Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Manca Žerovnik

Odkrivanje vzorcev v večglasni glasbi z nenadzorovanim učenjem

magistrsko delo

magistrski program druge stopnje
računalništvo in informatika

Mentor: doc. dr. Matija Marolt

Ljubljana, 2017

Zahvala

Hvala doc. dr. Matiji Maroltu za odlično mentorstvo, ki je močno pripomoglo k nastanku dela. Najlepša hvala asistentu Matevžu Pesku za navdušenje, spodbude, pomoč in sodelovanje. Hvala tudi ostalim članom Laboratorija za računalniško grafiko in multimedije. Hvala Jeri za lektoriranje.
Posebna zahvala gre moji družini in Primožu, ki so me ves čas podpirali pri študiju in tudi na vseh ostalih področjih mojega življenja. Hvala prijateljem in vsem, ki lepšate moje življenje.

Manca Žerovnik, 2017

Vsem dobrim vzorcem, da bi se še naprej ponavljali.

Kazalo

Povzetek

Naslov: Odkrivanje vzorcev v večglasni glasbi z nenadzorovanim učenjem

V delu predstavimo nadgradnjo kompozicionalnega hierarhičnega modela za odkrivanje ponavljajočih vzorcev v večglasni glasbi v simbolični obliki. Kompozicionalni hierarhični model je globoka arhitektura zgrajena iz več nivojev, ki vzorce išče na podlagi nenadzorovanega učenja. Algoritem preizkusimo na javno dostopnih zbirkah in ga primerjamo z ostalimi algoritmi na tem področju. Predstavimo upodobitev rezultatov modela v obliki spletne aplikacije. Na podlagi najdenih ponavljajočih vzorcev naredimo klasifikacijo slovenskih ljudskih pesmi glede na variantni tip.

Ključne besede

pridobivanje informacij iz glasbe, kompozicionalni hierarhični model, odkrivanje ponavljajočih vzorcev

Abstract

Title: Discovery of patterns in polyphonic music in an unsupervised manner

In this work we introduce an upgrade of compositional hierarchical model for repated pattern discovery in polyphonic symbolic music. Compositional hierarchical model is a deep architecture with multi layer structure. Pattern discovery is made in an unsupervised manner. We test algorithm on public datasets and compare it with other approaches. We present a visualization of results in form of web application. Finally we make a classification of slovenian folk songs based on discovered repeated patterns.

Keywords

music information retrieval, compositional hierarchical model, discovery of repeated patterns

Poglavje 1

Uvod

Glasba je v naši družbi zelo močno prisotna. Skoraj vsak posameznik se srečuje s poslušanjem, veliko ljudi pa tudi z ustvarjanjem in analizo glasbe. Kljub pomembnosti glasbe, ki se v družbi kaže, pa je pridobivanje informacij iz glasbe (ang. Music Information Retrieval - MIR) mlado področje, ki se je začelo razvijati manj kot tri desetletja nazaj. Gre za interdisciplinarno področje, ki združuje strojno učenje, procesiranje signalov, muzikologijo, glasbo in psihologijo. S pomočjo tehnologije in znanj iz teh ved želimo iz glasbe izluščiti človeku koristne informacije. Od svojega začetka naprej se področje hitro razvija. Glavni razlogi za hiter razvoj so uspešno stiskanje zvoka v poznih devetdesetih, povečanje moči osebnih računalnikov in s tem možnost pridobivanja informacij v smiselnem času, povečanje razširjenosti prenosnih predvajalnikov glasbe in v zadnjem času pojav pretočnih storitev za glasbo kot so Spotify, Grooveshark, Rdio, Deezer idr., ki omogočajo neomejen dostop do glasbe kjerkoli in kadarkoli [1]. Robusten sistem za pridobivanje informacij iz glasbe zaradi kompleksnosti prepletanja različnih vidikov glasbe še ne obstaja, bi pa bil zelo zaželjen in bi prinesel veliko dodano vrednost obstoječim glasbenim zbirkam [2]. Za pridobitev takšnega sistema se je potrebno ukvarjati s posameznimi problemi, ki glasbo pretvarjajo v druge oblike informacij kot so ocenjevanje osnovnih frekvenc, zaznavanje tempa, ločevanje glasov in instrumentov, izločitev glavne melodije, zaznavanje ritma, zaznavanje akor-

dov, iskanje vzorcev itd.

Problem, ki je ključen za razumevanje in interpretacijo glasbe, je pridobivanje ponavljajočih vzorcev in tem [3] v glasbi, s čimer se ukvarjamo v tem delu. Rezultate odkrivanja vzorcev lahko uporabimo na primer za analizo zbirk skladb, za iskanje povezav med vzorci različnih avtorjev, izvorov ali za izboljšano navigacijo po zbirkah. S problemom se v zadnjem času ukvarja vse več raziskovalcev ([4], [5], [6], [7]). Od leta 2013 je vključen v dogodek za skupno evalvacijo algoritmov pridobivanja informacij iz glasbe MIREX (ang. Music Information Retrieval Evaluation Exchange) [8].

Ponavljajoči vzorec v glasbi je zaporedje tonskih višin s pripadajočimi časi, ki se vsaj enkrat ponovi in ni nujno povsem enak prvemu. Vzorec, ki se ponovi, je večinoma časovno zamaknjen in transponiran ter je variacija prvega vzorca (ni nujno identičen prvemu). Vzorec je lahko vsebovan v drugem vzorcu. Ni vsako ponavljanje vzorec, ki ga iščemo. Nekatere znane skladbe imajo vzorce, ki se ponavljajo, in so dolgi le 4 tone (npr. uvod skladbe Purple Haze Jimmyja Hendrixa). To pa ne pomeni, da so vsa ponavljanja 4-ih tonov glasbeno pomemben ponavljajoči vzorec. Iščemo le zaznavno in analitično pomembne vzorce. Algoritmi za reševanje problema morajo zato vsebovati ne samo iskanje ponavljanj zaporedij in njihovih variacij ampak tudi prepoznavanje pomembnosti vzorcev (npr. izločanje vzorcev, ki imajo veliko verjetnost, da so se pojavili naključno).

Probleme, ki se pojavljajo na področju pridobivanja informacij iz glasbe, lahko rešujemo na različnih vhodnih predstavitvah. Glede na vrsto zapisa razlikujemo med zvočnim zapisom in simboličnimi zapisi (note, zapis MIDI ...). Prav tako lahko glasbo ločujemo na podlagi stopnje polifonije. Pri enoglasni glasbi so tipično rešitve problemov bolj enostavne, večglasna glasba pa je kompleksnejša, kar se odraža tudi pri pristopih pridobivanja informacij iz glasbe. V našem delu smo se omejili na iskanje vzorcev v večglasni glasbi v simboličnem zapisu.

Cilj naloge je nadgradnja hierarhičnega kompozicionalnega modela [9], ki je bil uporabljen za iskanje vzorcev v enoglasni glasbi [6], [10], na iskanje v

večglasni glasbi. Rezultate dela bomo za lažjo interpretacijo upodobili in jih primerjali z rezultati drugih raziskovalcev, ki rešujejo isti problem. Rezultate modela bomo uporabili pri analizi glasbene zbirke slovenskih ljudskih pesmi.

V delu najprej bolj natančno predstavimo področje in sorodna dela. Nato predstavimo kompozicionalni hierarhični model, njegove prilagoditve in izdelane upodobitve rezultatov. V poglavju 5 predstavimo rezultate modela, jih komentiramo in primerjamo z ostalimi, v poglavju 6 predstavimo uporabo modela na zbirki slovenskih ljudskih pesmi. Na koncu v poglavju 7 strnemo ugotovitve in predstavimo možne izboljšave.

Poglavje 2

Pregled področja

2.1 Pridobivanje informacij iz glasbe

Pridobivanje informacij iz glasbe se ukvarja z analizo, manipulacijo in ustvarjanjem glasbe s pomočjo računalnika. Schedl [1] je leta 2014 definiral štiri glavna podpodročja razvoja:

- Prepoznavanje značilnic - to je podpodročje, na katerem temeljijo vsa ostala. Gre za prepoznavanje barve, melodije, osnovnih frekvenc, ritma, tempa, vzorcev, segmentov, akordov, tonalitete, akordov, pojavitve dogodkov in analizo strukture v glasbi.

- Podobnost - tukaj se razvijajo predvsem mere podobnosti v glasbi, prepoznavanje priredb skladb in poizvedovanje z mrmranjem.

- Klasifikacija - v tem podpodročju se razvija prepoznavanje čustev in razpoloženja v glasbi, prepoznavanje žanra, klasifikacija instrumentov, prepoznavanje avtorjev, umetnikov in pevcev ter samodejno označevanje.

- Aplikacije - to podpodročje se ukvarja z uporabniško usmerjenimi aplikacijami za poizvedovanje in pridobivanje glasbe glede na vsebino, uporabo priporočilnih sistemov za glasbo, tvorjenje seznamov predvajanja, poravnavanje zvoka z notami in sinhronizacijo, ocenjevanje popularno-

sti skladb ali avtorjev, upodabljanje glasbe, iskanje po glasbi, interakcijo z glasbo, ukvarja se tudi s personaliriziranimi adaptivnimi sistemi.

Informacije v glasbi imajo na najnižjem nivoju sedem vidikov [2]: vidik intonacije oz. višine tona, časovni vidik, vidik harmonije, vidik barve zvoka, uredniški vidik, tekstualen vidik in bibliografski vidik. Vidiki se združujejo in prepletajo. Tako sestavljajo kompleksnejše informacije. Mi se bomo ukvarjali z iskanjem ponavljajočih vzorcev, ki več vidikov združuje v višjenivojsko informacijo. Po zgornji razdelitvi na podpodročja problem iskanja ponavljajočih vzorcev spada na podpodročje podobnosti.

Pridobivanje informacij je zahtevna naloga, ne le zaradi vseh različnih vidikov informacij v glasbi ampak so tu še izzivi interdisciplinarnosti - ker je potrebno sodelovati z drugimi področji; izzivi multikulturnosti - ker se različne kulture različno glasbeno izražajo; in pa različnost v odzivanju in interakciji posameznikov v odnosu do glasbe [2]. Pri tem je seveda za nas najbolj relevantno vprašanje, kaj je, ne glede na kulturni izvor in različnost v odzivih posameznikov, ponavljajoči vzorec v glasbi in ali vsi ljudje zaznavamo iste dele kot ponavljajoči vzorec v glasbi.

2.2 MIREX

MIREX je kratica za Music Information Retrieval Evaluation eXchange, kar v slovenščini pomeni izmenjava vrednotenj pridobivanja informacij iz glasbe. Gre za dogodek, ki se vsako leto izvede v kombinaciji s konferenco ISMIR, ki je na tem področju vodilna konferenca. MIREX formalno definira probleme in vrednotenja za algoritme, nato se tu vsako leto primerjajo najsodobnejši sistemi, algoritmi in tehnike pridobivanja informacij iz glasbe. Vodi ga mednarodni laboratorij za pridobivanje informacij iz glasbe IMIRSEL (nternational Music Information Retrieval Systems Evaluation Laboratory) iz Univerze Illinois (ang. University of Illinois at Urbana-Champaign).

2.3 Pridobivanje ponavljajočih vzorcev

Pridobivanje ponavljajočih vzorcev obravnavamo na način, ki je definiran v okviru evalvacije MIREX [11]. Razvili bomo algoritem, ki sprejme skladbo, vrne pa seznam ponavljajočih vzorcev. Ponavljajoči vzorec je množica parov pojavitvenega časa in tona, ki se v skladbi pojavi vsaj dvakrat, torej se vsaj enkrat ponovi. Vzorec, ki se ponovi, je lahko enak prvemu ali pa je variacija prvega vzorca. Lahko je časovno zamaknjen, lahko pa se prekriva s prvim vzorcem ali pa je v njem vsebovan. Vzorci niso vnaprej definirani, algoritem mora sam poiskati relevantne vzorce. Pri tem ne iščemo vseh ponavljajočih vzorcev, ampak le tiste, ki so vsebinsko pomembni. Algoritem evalviramo na nalogi iskanja vzorcev znotraj ene skladbe (ang. intra-opus discovery), kar pomeni, da je na vhodu algoritma ena sama skladba, izhod algoritma pa so vzorci v tej skladbi. Pristop, ki ga razvijamo, je sicer splošen in podpira tudi iskanje vzorcev v množici skladb (ang. inter-opus discovery), a je zaenkrat na voljo premalo ekspertnih anotacij, da bi se lahko izvedli evalvacijo.

2.4 Pregled sorodnih del

V splošnem se algoritmi za odkrivanje ponavljajočih vzorcev in tem v glasbi delijo na tiste, ki delujejo na enoglasni glasbi, in na tiste bolj kompleksne, ki se ukvarjajo z večglasno glasbo. Pristope lahko razdelimo na tiste, ki problem rešujejo z geometričnim pristopom, in na tiste, ki temeljijo na operacijah nizov [12]. Slednji se bolje izkažejo na

pri enoglasni glasbi ali pri večglasni glasbi, ki je zapisana po posameznih glasovih, medtem ko se pri večglasni glasbi bolje izkažejo geometrične metode, ker lahko upoštevajo več dimenzij glasbe. Algoritmi, ki temeljijo na operacijah nizov, večinoma glasbo najprej pretvorijo v niz tonskih višin ali intervalov, kar pomeni, da upoštevamo le eno glasbeno dimenzijo - če vhodno skladbo pretvorimo v niz tonskih višin, ritma, pojavitvenih časov not in ostalih informacij ne upoštevamo.

Algoritmov, ki so bili predstavljeni pred letom 2013, ne moremo neposre-

dno primerjati med seboj, ker je bilo prvo MIREX vrednotenje izvedeno šele leta 2013 in je vsak algoritem ocenjen drugače. Kljub temu bomo naredili kratek pregled vseh pristopov.

2.4.1 Algoritmi, ki temeljijo na operacijah z nizi

Hsu idr. [13] v glasbi odkrivajo netrivialne vzorce. To so vzorci tonov, ki se brez variacij večkrat ponovijo. Vzorci, ki so v celoti vsebovani v drugem vzorcu, se ne upoštevajo. Predstavijo dve metodi in jih med seboj primerjajo glede na hitrost delovanja. Pri prvi skladbo pretvorijo v niz imen not. Problem se pretvori v iskanje ponavljajočih podnizov v nizu. To storijo s pomočjo korelacijske matrike. Drugi pristop pa uporablja operacijo združevanja nizov in strukturo imenovano RP-drevo. Rezultate za dokaz učinkovitosti analizirajo in opišejo. Algoritem deluje na večglasni glasbi le, če je ločena na glasove.

Knopke [14] se je problema lotil na zbirki 101 maše skladatelja Palestrine. Gre za dela iz 16. stoletja. Algoritem dobro deluje na tej podmnožici vendar se osredotoča na posebne lastnosti tega obdobja in skladatelja, kar bi pri ostalih glasbenih delih povzročilo slabše delovanje. Osnovni princip delovanja je priponska tabela, ki je alternativa priponskemu drevesu, ki se pogosto uporablja pri iskanju ponavljajočih vzorcev v enoglasni glasbi [15]. Algoritem ni bil preverjen na podlagi anotacij, ampak analitično na podlagi rezultatov in muzikološkega znanja. Pomagal je pri analizi del Palestrine.

Chiu [16] je predstavil dva algoritma za iskanje vzorcev v večglasni glasbi. Ne osredotoča se na vsebino najdenega, ampak samo išče vzorce brez variacij, ki se ponavljajo in pri tem ocenjuje hitrosti obeh algoritmov. Boljši od obeh algoritmov deluje na podlagi leksikografskega drevesa.

Meek [17] predstavi algoritem za iskanje glavnih tem v glasbi, ki je zapisana v simboličnem zapisu. Deluje na principu iskanj ponavljanj. Algoritem je zanimiv, saj ne išče samo ponavljanj, ampak tako kot se problem kasneje izoblikuje, išče zaznavno pomembna ponavljanja. Osnovni princip delovanja algoritma je pridobitev vseh kombinacij not v skladbi od dolžine 2 do neke

konstante, potem pa iskanje tistih, ki se največkrat ponovijo. Zadnji in zelo pomemben korak je iskanje pomembnih tem med najdenimi ponavljajočimi vzorci. Algoritem je bil preizkušen na zbirki 60-ih del iz baroka, klasicizma, romantike in sodobnosti. V 98-ih odstotkih je algoritem odkril vzorce, ki so bili označeni kot 1. tema v ekspertni anotaciji Barlow-a [18], čigar anotacije so uporabili tudi pri zbirki, ki se uporablja za ocenjevanje problema odkrivanja ponavljajočih vzorcev na MIREX-u. Algoritem sicer kot vhod vzame večglasno glasbo, vendar na začetku izlušči najvišje zveneči zvok in na tem išče ponavljanja.

Conklin [19] glasbo predstavi kot več tokov, kjer vsak predstavlja določen vidik glasbe. Na ta način premošča problem upoštevanja le ene dimenzije podatkov, ki se pojavlja pri algoritmih, ki temeljijo na operacijah z nizi. Na podlagi tega z Markovim modelom poišče tiste dele glasbe, ki se pojavijo večkrat, kot je pričakovano. Pristop je preizkusil na 185-ih Bachovih koralih, kjer je uspešno našel nekatere glavne teme. Deluje na enoglasni glasbi.

2.4.2 Geometrične metode

Meredith že leta 2002 [20] predstavi geometrično metodo, ki jo kasneje še nadgradi [4] in preizkusi na MIREX-u. Analizirano glasbo predstavi kot večdimenzionalno množico točk. Med točkami geometrijsko poišče ponavljajoče vzorce in izbere najbolj pomembne. Vzorec je definiran kot maksimalni translacijski vzorec - MTP *(ang. maximal translational pattern)*. MTP je za vektor v definiran tako:

$$MTP(v, D) = \{p \,|\, p \in D \wedge p + v \in D\}, \tag{2.1}$$

kjer je D množica točk, ki predstavlja skladbo, $p + v$ pa označuje vzorec, ki obstaja, če vektor p transliramo za vektor v. Za predstavitev pojavitev istega vzorca definira translacijske ekvivalenčne razrede - TEC *(ang. translational equivalence classes)*. TEC vzorca P v množici točk D vsebuje tiste vzorce, ki so translacijsko ekvivalentni vzorcu P. Meredith predstavi več algoritmov, ki izmed najdenih vzorcev izberejo pomembne vzorce. Trenutno je to, če

se navezujemo na MIREX, najboljši pristop, ki deluje na večglasni glasbi, ki ni ločena na posamezne glasove. Najboljši izmed njegovih algoritmov, ki so po delovanju drug drugemu v osnovi zelo podobni, je SIATECCompres. Vsi algoritmi temeljijo na algoritmu SIA, ki v dani množici točk poišče vse MTP-je. Algoritem SIATEC izračuna vsem MTP-jem pripadajoče ekvivalenčne razrede. SIATECCompress v množici rezultatov T algoritma SIATEC poskuša poiskati podmnožico E, ki pokrije celotne vhodne podatke in pri tem maksimizira stisljivost te množice.

Tudi Collins [12] v večglasni glasbi v simboličnem zapisu išče translacijske ponavljajoče vzorce na podlagi posodobitve Meredithovega geometričnega algoritma SIA. Algoritem se imenuje SIACT in naslavlja problem "izoliranega članstva"v delovanju algoritma SIA.

2.4.3 Ostali pristopi

Peeters [21] je predstavil metodo, ki iz zvočnih posnetkov najprej izlušči značilnice povezane z ritmom in barvo zvoka - 13 melodično frekvenčnih kepstralnih koeficientov, 12 spektralno kontrastnih koeficientov in 12 Harmonic pitch class profiles (HPCP) koeficientov. Iz teh koeficientov z metodo iskanja glavnih komponent izbere komponente, ki pokrijejo več kot deset odstotkov variance. Potem pa iz teh komponent z uporabo Evklidske razdalje zgradi 3 podobnostne matrike. Vse skupaj poveže v podobnostne matrike drugega in tretjega reda. S pristopom ocenjevanja največje verjetnosti med izbranimi segmenti izbere tiste, ki so najbolj pomembni. Na koncu metodo na delu podatkovne zbirke MPEG7, ki je označen z "melody repetition", oceni za uspešno.

Nekoliko slabše deluje algoritem [22], ki na podlagi zvočnega zapisa zgradi transpozicijsko invariantno samopodobnostno matriko, iz katere s požrešnim pristopom poišče ponavljanja (diagonalne poti).

Pri iskanju vzorcev v enoglasni glasbi se uporablja iskanje ponavljajočih vzorcev v nizu s pomočjo priponskega drevesa [15]. Pomanjkljivost metode je, da ni bila preverjena, tako da bi jo lahko primerjali z drugimi. K problemu se

pristopa tudi z računsko metodo, ki temelji na uporabi valčne transformacije [5].

V [6] in [10] avtorji za enoglasno glasbo v simboličnem zapisu uspešno uporabijo kompozicionalni hierarhični model, ki z nenadzorovanim učenjem zgradi model, ki odraža najdene vzorce in teme in ga bomo za osnovo našega pristopa uporabili v našem delu.

Poglavje 3

Kompozicionalni hierarhični model za pridobivanje vzorcev iz večglasne glasbe

3.1 Kompozicionalni hierarhični model

Kompozicionalni hierarhični model *(ang. compositional hierarchical model - CHM)* je bil razvit leta 2007 za področje računalniškega vida [23]. Kasneje je bil preoblikovan za delovanje na področju glasbe [9]. Gre za biološko navdahnjen model, ki iz enostavnih konceptov gradi kompleksnejše, pri tem pa uporablja tudi mehanizma dodajanja manjkajočih informacij in odstranjevanja odvečnih informacij, ki sta značilna za človeško zaznavanje. Ime izhaja iz osnovnih lastnosti modela. Kompozicionalni, ker je zgrajen iz delov - vsak del predstavlja nek koncept, ki se preslika na vhodno informacijo. Hierarhični, ker se deli med sabo povezujejo v hierarhijo. Model je bil preizkušen na različnih opravilih pridobivanja informacij iz glasbe. Leta 2012 je bil uporabljen za [24] prepoznavanje akordov, leta 2014 za ocenjevanje osnovnih frekvenc [25], leta 2016 pa za transkripcijo glasbe [26] in prepoznavanje ponavljajočih vzorcev v enoglasni glasbi [27]. Model deluje tako, da na podlagi vhodnih podatkov zgradi hierarhijo gradnikov, ki jih imenujemo deli

(ang. parts), ki vhodno informacijo hierarhično predstavijo. Ko imamo hierarhijo naučeno, lahko na njej izvedemo inferenco, kar pomeni da hierarhija sprejme nove vhodne podatke, kot izhod pa nam vrne množico aktiviranih delov hierarhije, ki nam predstavljajo neko informacijo o vhodnih podatkih. V nadaljevanju bomo predstavili model, ki smo ga uporabili za pridobivanje vzorcev iz večglasne glasbe. Ob tem predstavimo splošen koncept delovanja modela in narejene prilagoditve, ki poleg delovanja na večglasni glasbi vključujejo tudi hitrostno optimizacijo.

3.2 Vhodni podatki

Vhod v model je v našem primeru katerakoli večglasna ali enoglasna skladba zapisana v simboličnem zapisu, konkretno uporabljamo zapis MIDI. Na vhod modela pošljemo tekstovno datoteko, ki je sestavljena po principih evalvacije MIREX, in kjer vsaka vrstica predstavlja ton v skladbi, sestavljajo pa jo z vejico ločeni podatki. Vhod lahko definiramo kot množico seznamov:

$$\mathcal{S} = \{[N_o, N_{p1}, N_{p2}, N_d, N_s]\}, \tag{3.1}$$

ki predstavljajo:

- N_o *(ang. onset time)* - to je pojavitveni čas nastopa tega tona, ki se meri od ničle v udarcih četrtinke.
- N_{p1} *(ang. MIDI note number)* - višina tona, ki je predstavljena v MIDI formatu, ki predstavlja preslikavo klavirskih tipk v številke.
- N_{p2} *(ang. morphetic pitch number)* [28] - višina tona, zapisana v morfetični vrednosti, to je prav tako višina pretvorjena v celo število in sicer se ta določi na podlagi mesta note v notnem črtovju v odvisnosti od ključa. Zapis ignorira predznake - višaje, nižaje in razvezaje. Če torej definiramo, da je številka tona B1 enaka 0, so morfetične vrednosti B1#, C2, C2# enake 0, 1, 1.

- N_d - dolžina tona merjena v četrtinkah
- N_s - številka notnega črtovja - te vrednosti ne uporabljamo.

3.3 Zgradba modela

3.3.1 Hierarhija

Osnovni gradniki modela so deli. Deli se povezujejo v hierarhijo. Na neki vhodni množici skladb se tako zgradi celotna hierarhija z nivoji $L_0, ..., L_n$, vsak nivo L_m pa vsebuje dele $P_0^m, ..., P_i^m$. Vsak nivo predstavlja kompleksnejše koncepte, v našem primeru so to glasbeni vzorci. Nivoji niso skriti, struktura delov je eksplicitna in tovrstna transparentnost nam pomaga pri razumevanju delovanja modela ter analiziranju glasbenih del.

3.3.2 Del

Deli, torej osnovni gradniki modela, predstavljajo sopojavitve tonskih višin v vhodnih podatkih. Za tonske višine smo tam, kjer smo imeli na voljo podatek o morfetični vrednosti, uporabili te, drugje smo uporabili MIDI vrednosti. V nadaljevanju bomo za obe vrednosti uporabljali oznako N_p.

Nivo L_0

Na nivoju L_0 imamo en sam del P_1^0, ki predstavlja pojavitev tonske višine na vhodu.

Nivo L_n; $n > 1$

Del na nivoju L_n; $n > 1$ je sestavljen iz delov na nivoju $n - 1$:

$$P_m^n = \{P_{k0}^{n-1}, P_{k1}^{n-1}(\mu)\}, \tag{3.2}$$

kjer μ predstavlja razdaljo med poddeloma. Tak del vsebuje naslednje podatke:

- ID - identifikacijska številka dela na tem nivoju.
- T = $[T_{k0}^{n-1}, T_{k1}^{n-1}]$ - seznam razdalj med tonskimi višinami - združitev seznamov poddelov iz katerih je del sestavljen. Ta seznam predstavlja relativne razdalje med tonskimi višinami in si ga lahko predstavljamo kot vzorec, ki se v skladbi lahko večkrat aktivira.
- k_0 - centralni del, ki ga predstavlja identifikacijska številka prvega dela iz prejšnjega nivoja, iz katerega je nastal ta del (P_{k0}^{n-1}).
- (P_{k0}^{n-1}, P_{k1}^{n-1}) - par delov prejšnjega nivoja iz katerih je nastal ta del.
- μ - razdalja med poddeloma in sicer med prvim tonom P_{k0}^{n-1} in prvim tonom P_{k1}^{n-1}.

Primer hierarhije z deli vidimo na sliki 3.1.

3.3.3 Aktivacija

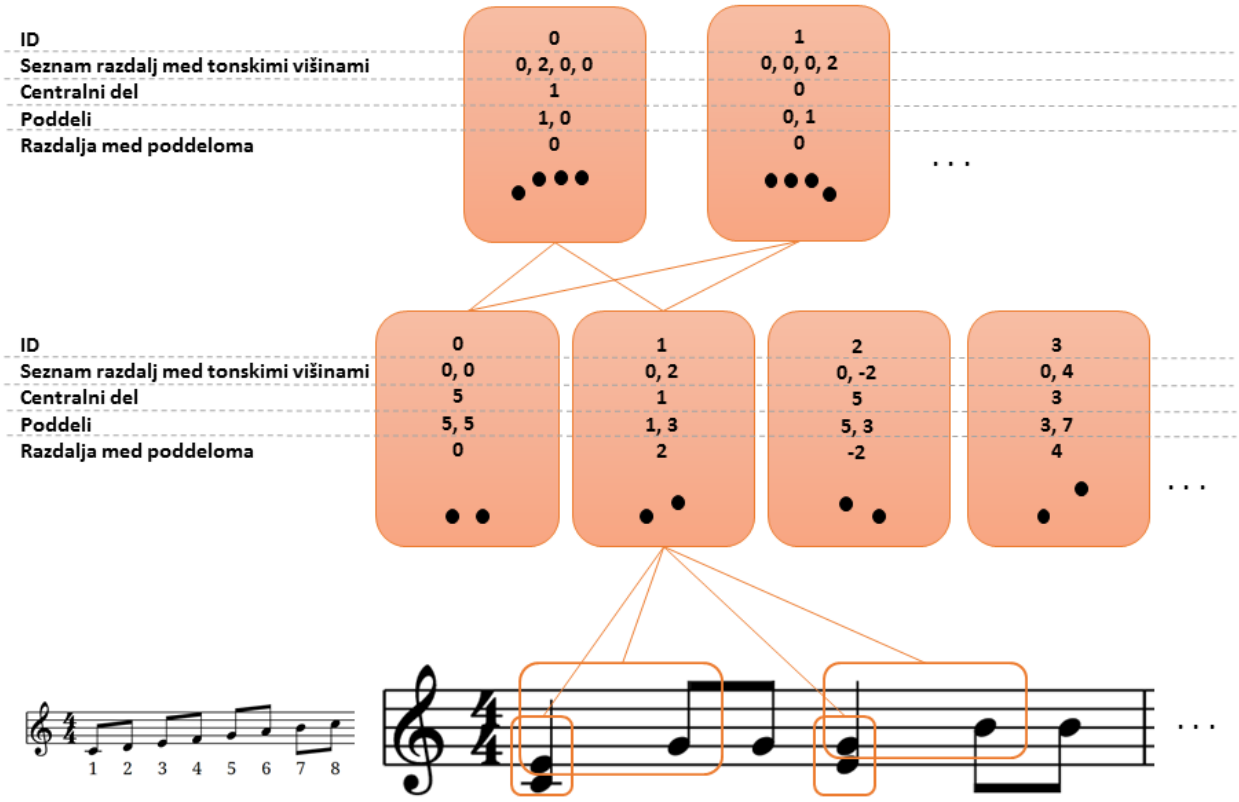
Delom v fazi inference pripisujemo aktivacije. Aktivacija pomeni, da lahko koncept, ki ga del predstavlja (vzorec), preslikamo na vhodno skladbo, torej da obstaja na vhodu modela. Del se v skladbi na različnih lokacijah lahko aktivira večkrat, torej ima več aktivacij. Aktivacijo posameznega dela formalno lahko zapišemo kot trojček: $\langle A_l, A_t, A_M \rangle$, ki za nivo L_n, $n > 0$ predstavlja naslednje podatke:

- A_l predstavlja lokacijo, to pomeni tonsko višino, prvega tona, kjer se je del aktiviral. Na nivoju n, jo dobimo tako:

$$A_l(P_i^n) = A_l(P_{k0}^{n-1}) \tag{3.3}$$

- A_t predstavlja pojavitveni čas prvega tona, kjer se je del aktiviral. Na nivoju n ga dobimo tako:

$$A_t(P_i^n) = A_t(P_{k0}^{n-1}) \tag{3.4}$$



Slika 3.1: Na sliki je prikazana hierarhija modela zgrajena iz notnega zapisa, ki je prikazan na dnu slike. Za skico smo notam pripisali morfetične vrednosti od 1 do 8, tako kot je prikazano spodaj levo. Rdeči pravokotniki predstavljajo dele, tako kot je opisano v 3.3.2. Prva vrsta predstavlja nivo L_1, druga pa nivo L_2. Nivo L_0 predstavlja notni zapis. Na sliki se vidi, kako se deli med seboj združujejo. S povezavami smo prikazali, od kod pride del nivoja L_1 z identifikacijsko številko 1, in iz katerih delov nastaneta dela drugega nivoja. Nov del na višjem nivoju ima lahko za poddela tudi isti del, če se ta sproži na dveh različnih lokacijah v vhodnih podatkih. Po enakem principu se gradijo tudi višji nivoji.

- A_M predstavlja magnitudo, ki določa jakost aktivacije. Magnitudo izračunamo kot uteženo vsoto magnitud aktiviranih poddelov znotraj časovnega okna, ki z nivoji raste:

$$A_M(P_i^n) = \tanh\left(\frac{1}{K}\sum_{j=0}^{K-1} w_j A_M(P_{k_j}^{n-1})\right) \tag{3.5}$$

$$w_j = \begin{cases} 1: & \delta_{Tj} < \tau_W \\ 0: & \delta_{Tj} \geq \tau_W \end{cases} \tag{3.6}$$

$$\delta_{Tj} = A_t(P_{k_j}^{n-1}) - A_t(P_{k_0}^{n-1}) \tag{3.7}$$

$$\tau_W = \min(2^{(n+1)}, t_l), \tag{3.8}$$

kjer t_l predstavlja zadnji pojavitveni čas v skladbi.

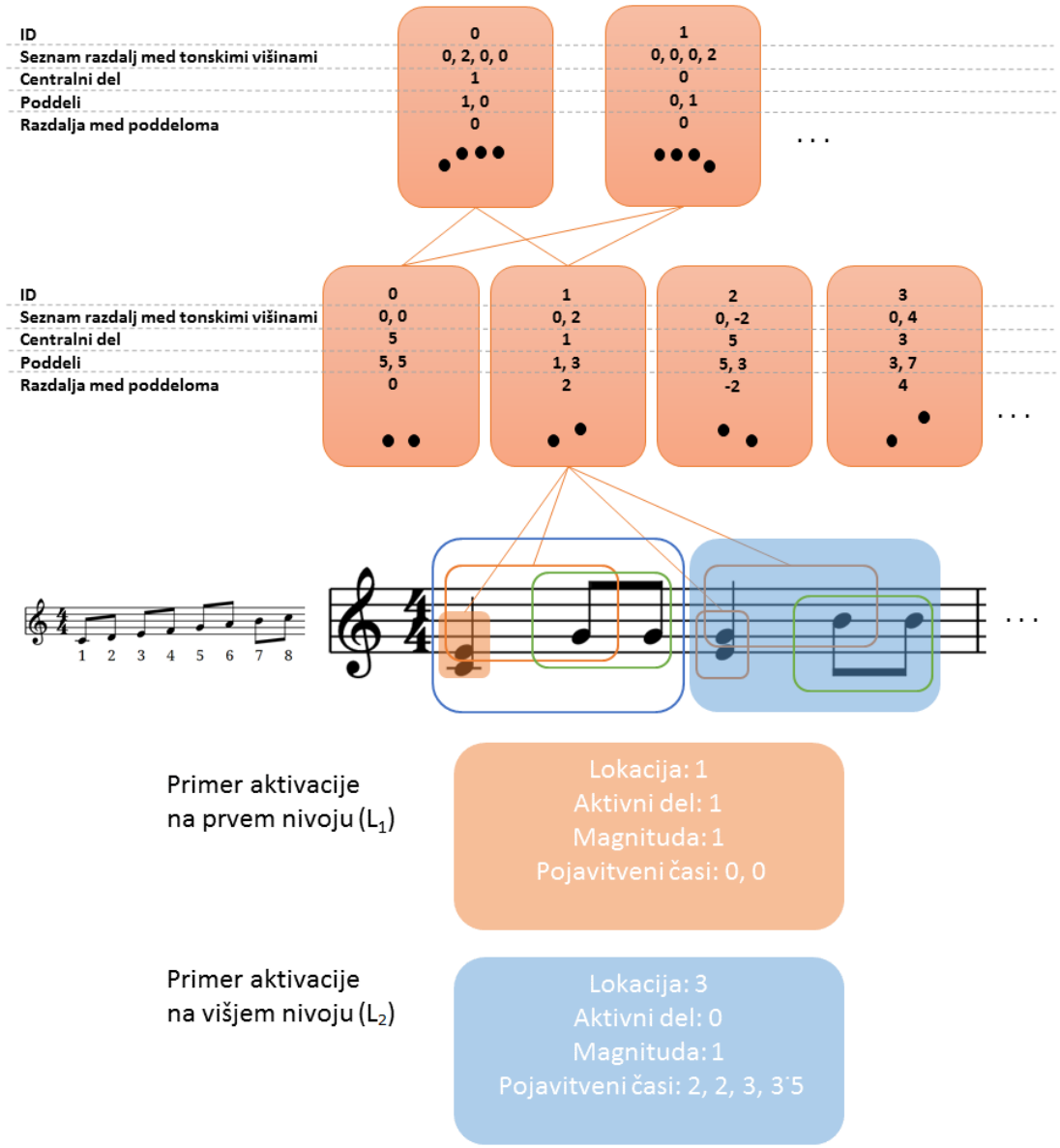
Vrednosti A_l, A_t in A_M torej propagirajo vrednosti centralnega dela po hierarhiji navzgor. Na ta način lahko sledimo aktivacijam tudi od zgoraj navzdol. Na nivoju L_0 imamo samo en del P_1^0, ki se aktivira za vse vhodne vrednosti v $\mathcal{S}$. Aktivacije za del P_1^0 dobimo kot:

$$\langle A_l, A_t, A_M \rangle = \langle N_p, N_0, 1 \rangle \tag{3.9}$$

Primer aktivacije je prikazan na sliki 3.2.

3.4 Učenje modela

Učenje modela je nenadzorovano. Za problem odkrivanja ponavljajočih vzorcev je to nujno, ker je že narava problema (odkrivanje vzorcev v glasbenih



Slika 3.2: Na sliki je enak primer kot na sliki 3.1. Dodali smo aktivacije delov za dela 0 in 1 iz prvega nivoja in za del 0 iz drugega nivoja. Aktivacije delov so označene z okvirji v notnem zapisu. Oranžen okvir označuje aktivacije dela 1 iz prvega nivoja, zelen okvir prikazuje aktivacije dela 0 iz prvega nivoja, moder okvir pa predstavlja aktivacije dela 0 iz drugega nivoja. Okvira, ki sta zapolnjena z barvo, sta aktivaciji, ki sta spodaj natančneje opisani z atributi, ki jih vsaka aktivacija v modelu vsebuje.

delih) taka, da se vseh vzorcev ni mogoče naučiti, prav tako pa ne obstajajo velike anotirane zbirke, na katerih bi lahko (tudi enostavnejše) modele učili. Učenje poteka po nivojih, tako da najprej zgradimo prvi nivo hierarhije, potem pa iz vsakega prejšnjega nivoja gradimo naslednje. Psevdokoda učenja je opisana v algoritmu 1, posamezne metode algoritma pa v nadaljevanju.

Algoritem 1 Psevdokoda učenja

```
for stevecNivojev = 0; stevecNivojev < steviloNivojev; stevecNivojev++ do
    Inferenca();
    Naredi kandidate za dele();
    Izberi Dele();
end for
```

3.4.1 Pridobivanje kandidatov za dele

Učenje začnemo z ustvarjanjem kandidatov za dele. Za nivo n to poteka tako, da zgradimo histogram sopojavitev vseh kombinacij aktivacij iz nivoja $n-1$, ki imajo magnitudo večjo od 0, $A_M > 0$. Histogram normaliziramo, kandidate za gradnike pa nato generiramo iz vseh sopojavitev, ki presežejo učni prag γ, tako da iz delov, ki pripadajo izbranim aktivacijam, ustvarimo nov del. Rezultat postopka je množica kandidatov za dele $\mathcal{K}$.

3.4.2 Izbira delov

V prvem delu učenja posameznega nivoja pridobimo množico kandidatov za dele. Ker je teh veliko in so lahko redundantni, njihovo število zmanjšamo s požrešno metodo, ki iterativno izbira dele. V vsaki iteraciji izbere del, ki, če ga dodamo v množico že izbranih delov, prispeva največ k *pokritju* vhodnega signala. Izbrani del dodamo v množico izbranih delov in odstranimo iz množice kandidatov. Pokritje, ki ga uporabljamo kot mero za izbor delov, je preslikava posameznega dela na nivo L_0 in nam pove, koliko vhodne informacije množica delov pokrije. Bolj formalno lahko pokritje dela C definiramo

kot unijo pokritja poddelov na nižjem nivoju:

$$C(A(P_i^n)) = \bigcup_{j=0}^{K-1} C(A(P_{k_j}^{n-1})) \tag{3.10}$$

Na nivoju L_0 je pokritje dogodka definirano kot A_l aktivacije dogodka (3.11):

$$C(A(P_1^0)) = \begin{cases} A_l(P_1^0): & A_M(P_1^0) > 0 \\ \emptyset: & sicer \end{cases}. \tag{3.11}$$

Računanje pokritja je časovno najbolj potraten del učenja, zato smo ga paralelizirali. Iteracije izbiranja delov potekajo toliko časa, dokler za množico kandidatov $\mathcal{K}$ in množico izbranih delov $\mathcal{I}$ ni izpolnjen eden od naslednjih pogojev:

- $\mathcal{K} = \varnothing$
- $C(\mathcal{I}) > \omega$, trenutni odstotek pokritja vhodnega signala z izbranimi deli je večji od parametra ω, ki določa prag pokritja za dodajanje kandidatov.
- $(C(\mathcal{I} \cup P_{new}) - C(\mathcal{I})) < \phi$, doprinos novega dela k pokritju je manjši od parametra ϕ. Ko enkrat začnemo dodajati dele, ki ne prinašajo več veliko nove informacije, zaključimo.

3.5 Inferenca

Inferenca je postopek računanja aktivacij delov za določen vhod modela. Deli, ki se aktivirajo, predstavljajo pojavitve naučenih vzorcev, ki se pojavijo v vhodnih podatkih. Na nižjih nivojih so to krajši vzorci, na višjih daljši. Del, ki se aktivira večkrat, je ponavljajoči vzorec. Osnovni princip inference je, da se nek del aktivira, če so aktivirani vsi njegovi poddeli. To pomeni, da pri inferenci poiščemo dele, ki imajo aktivirane vse poddele in jim pripišemo aktivacijo, tako kot je opisano v 3.3.3. Inferenca poteka po

nivojih od najnižjega do najvišjega. Pri postopku uporabljamo dva dodatna mehanizma, ki povečata robustnost modela in sta opisana v nadaljevanju.

3.5.1 Biološko navdahnjeni mehanizmi

Halucinacija

Halucinacija daje modelu zmožnost, da, tako kot človeški zaznavni sistemi, manjkajočo informacijo zapolni z najbolj logično razlago. To pri modelu pomeni, da se nek del lahko aktivira, tudi če niso aktivirani vsi njegovi poddeli oz. drugače povedano, aktivacija dela je možna tudi, če na vhodu manjka del koncepta (vzorca), ki ga predstavlja. Jakost halucinacije lahko nadziramo s parametrom τ_H, ki v primeru iskanja ponavljajočih vzorcev določa kakšen odstotek tonskih višin, ki jih del pokriva, mora obstajati v vhodnem signalu, da se del aktivira. Če je $\tau_H = 1$, model deluje enako kot bi deloval brez halucinacije, kar pomeni, da mora biti informacija, ki jo pokriva del, stoodstotno zastopana v vhodnem signalu. Manjša vrednost parametra povečuje število aktivacij, omogoča pa tudi iskanje variacij ponavljajočih vzorcev.

Inhibicija

Inhibicija deluje tako, da odstranjuje redundantne informacije. Pri učenju modela namreč pogosto nastanejo deli, ki hkrati pokrivajo enak del vhodnega signala (predstavljajo iste vzorce) in torej tekmujejo za njegovo pokritje. Z inhibicijo lahko zaviramo tovrstne redundantne aktivacije in sicer tako, da, ko dve aktivaciji delov z istega nivoja pokrivata isto vhodno informacijo, aktivacijo z manjšo jakostjo odstranimo. Odstotek informacije, ki mora biti enak, da se nek del inhibira, določa inhibicijski parameter τ_I. Aktivacija dela P_i^n se torej inhibira, kadar sta dosežena sledeča pogoja:

$$\exists \bigcup_{k=1}^{K} P_{j_k}^n : \frac{|C(A(P_i^n)) \setminus \bigcup_{k=1}^{K} C(A(P_{j_k}^n))|}{|C(A(P_i^n))|} < \tau_I \tag{3.12}$$

in

$$\forall P^n_{j_k} \in \bigcup_{k=1}^{K} P^n_{j_k} : A_M(P^n_{j_k}) > A_M(P^n_i). \tag{3.13}$$

kjer $C(A)$ predstavlja pokritje aktivacije (enačba 3.10), A_M magnitudo aktivacije, parameter τ_I pa uravnava jakost inhibicije in zavzema vrednosti med 0 in 1.

3.5.2 Iskanje ponavljajočih vzorcev

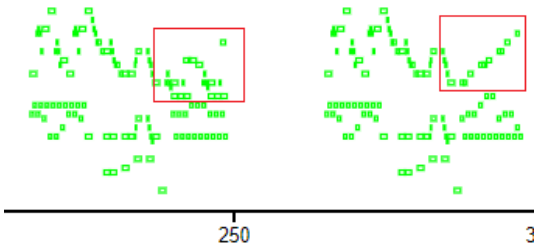
Iskanje vzorcev na neki skladbi lahko izvedemo z modelom naučenim na isti skladbi ali na širši glasbeni zbirki. Vzorce poiščemo z inferenco modela nad vhodno skladbo. S tem dobimo množico aktivacij delov. Da na podlagi aktivacij dobimo množico ponavljajočih vzorcev, aktivacije filtriramo. Najprej izberemo samo aktivacije delov, ki se aktivirajo vsaj dvakrat. Aktivacije delov, ki so podmnožica aktivnih delov na višjih nivojih, odstranimo, združimo tudi aktivaciji v primeru, da ena aktivacija pokriva vse dogodke druge aktivacije, tako da naredimo unijo pokritih dogodkov. Postopek izvedemo po posameznih nivojih.

3.5.3 Izhod modela

Izhod modela je množica vzorcev, ki se pojavijo v vhodni skladbi. Vsak vzorec je sestavljen iz najmanj dveh pojavitev. Vsaka pojavitev vzorca je zapisana kot zaporedje parov (pojavitveni čas, tonska višina). Primer vzorca z dvema pojavitvama je prikazan na sliki 3.3.

3.6 Problem večglasja

Model, ki smo ga vzeli za izhodišče našega dela, je bil razvit za iskanje vzorcev v monofonočnih skladbah. Model smo prilagodili tako, da deluje tudi na večglasnih vhodnih podatkih in ob tem vrača smiselne rezultate.



Slika 3.3: Na sliki je primer dveh pojavitev vzorca. Abscisna os predstavlja čas v udarcih četrtinke (prikazan je le določen izsek skladbe), ordinatna os pa predstavlja tonsko višino. Vsak pravokotnik predstavlja ton, ki se pojavi v skladbi. Vidimo, da smo našli dve pojavitvi vzorca, ki se med seboj malo razlikujeta. Razlika je vidna v delu, ki je označen z rdečim okvirjem.

Večglasne skladbe pri iskanju vzorcev predstavljajo težji problem, ker je možnosti za iskanje veliko več. Vzorec se lahko pojavi samo v eni liniji glasov v večglasju (horizontalno) ali pa zavzema vse glasove (vertikalno), vhodni podatki, ki jih dobimo, pa niso ločeni na posamezne glasove večglasja, tako da mora model tovrstne zakonitosti odkriti sam.

Osnovna predelava Za delovanje na večglasnih skladbah smo morali v osnovnem modelu najprej prilagoditi gradnjo novih delov. Osnovni model je dele generiral na osnovi indeksiranih zaporednih dogodkov, ker je predpostavljal, da dva dogodka ne moreta biti sočasna. Ker pri polifonični glasbi temu ni tako, smo morali pri učenju in inferenci dogodkom dodati časovno komponento. Spremenili smo pogoje, ki so dovoljevali gradnjo delov le iz zaporednih dogodkov. Poleg tega smo pri računanju pokritja delov zaradi prevelike časovne zahtevnosti odstranili inhibicijo in halucinacijo, tako da se ta izvede le v fazi inference.

Predelati smo morali kodo za zaključno filtriranje, tako da združevanje in računanje podobnosti med vzorci deluje tudi na večglasnih vzorcih. Dodali smo mehanizem, ki preveri podobnost med vsemi najdenimi vzorci. Pri-

merjamo vse pojavitve vzorcev. Če najdemo dve pojavitvi, ki imata oceno podobnosti večjo od parametra β, ki določa delež enakih dogodkov v dveh pojavitvah vzorca na različnih lokacijah, rečemo da gre za enak vzorec. Vzorca, ki vsebujeta opazovani pojavitvi, združimo. Znotraj novega vzorca pregledamo podobnost med pojavitvami. Če ocena podobnosti dveh pojavitev novega vzorca presega vrednost parametra β, krajšo od teh dveh pojavitev odstranimo. Uporabili smo vrednost $\beta = 0.8$. Primer rezultata, s filtriranjem in brez, je prikazan na sliki 3.4. Preizkusili smo več metod za ocenjevanje podobnosti vzorca in na koncu za oceno podobnosti uporabili Jaccardov indeks:

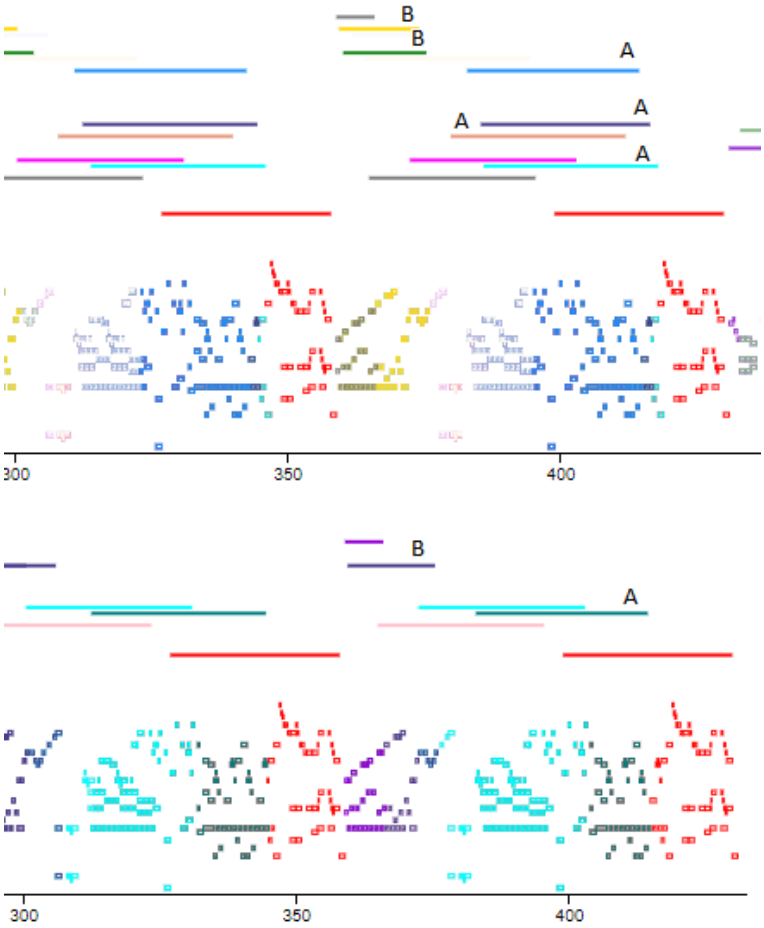
$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.14}$$

, kjer množici A in B predstavljata dogodke v posamezni pojavitvi vzorca.

Omejevanje aktivacij Pri učenju modela velik problem predstavlja časovna zahtevnost, saj v primeru prevelikega števila kandidatov za dele postane prepočasno.

Najbolj časovno zahtevna je metoda izbiranja delov, ki ima časovno zahtevnost $O(p^2 * a)$, kjer je p število kandidatov za dele, a pa število aktivacij prejšnjega nivoja. Število kandidatov za dele bi bilo brez omejitev parametrov na nivoju n enako $\frac{p_a(p_a-1)}{2}$, kjer je p_a število delov na nivoju $n-1$, ki imajo vsak eno aktivacijo.

Za optimizacijo postopka učenja smo prilagodili funkcijo računanja aktivacij, da uravnava število delov, ki se aktivirajo, kar pomeni, da se že pri samem učenju zgradi manj delov. V ta namen se je najbolje obnesla metoda, ki je zmanjšala magnitude aktivacij delov glede na število časovnih intervalov v vzorcih, ki ne vsebujejo dogodkov, merjenih v udarcih četrtinke, muzikološko to pomeni število dob v vzorcu, kjer je v vseh glasovih pavza. Princip smo izbrali na podlagi hipoteze, da zaporedje glasbenih dogodkov z več premori človek zazna kot več vzorcev. Ker v modelu del predstavlja koncept vzorca, v delih tako ne želimo prevelikih razmakov med pojavitvenimi časi dela, zato so taki deli nezaželeni.



Slika 3.4: Na sliki vidimo delovanje filtriranja vzorcev na primeru dela Mozartove skladbe. Zgornji graf prikazuje vzorce pred filtriranjem, spodnji pa po filtriranju. Abscisna os predstavlja čas v udarcih četrtinke, ordinatna vrednosti MIDI. Ista barva pomeni isti vzorec. Dogodki v vzorcih se prekrivajo, zato ne vidimo celotnih vzorcev, ker so lahko prekriti z drugimi. Črte nad grafi prikazujejo časovne intervale pojavitev vzorca, da lahko vidimo kje je vzorec, tudi če ga spodaj na grafu prekriva drug vzorec. Vidimo delovanje filtriranja, ko se vzorci, ki se najbolj prekrivajo, po filtriranju odstranijo. Na sliki smo označili primer filtriranja, tako da smo zraven časovnega intervala vzorca napisali črko. Vzorci, ki so na na zgornjem grafu označeni s črko A, prekrivajo velik odstotek enakih dogodkov, zato po filtriranju ohranimo samo vzorec A na spodnjem grafu. Enako velja za vzorce, označene s črko B.

Omejevanje števila vzorcev smo implementirali tako, da izračunamo število razmakov v vzorcu, ki so večji od enega udarca četrtinke, in to upoštevamo pri računanju magnitud aktivacij. Aktivacije smo morali za ta namen najprej razširiti, tako da poleg pojavitvenega časa A_t vsaka aktivacija nosi tudi informacijo o vseh pojavitvenih časih vzorca, ki se je aktiviral. Trojčku $\langle A_l, A_t, A_M \rangle$, ki predstavlja aktivacijo, smo dodali element A_T. Aktivacijo predstavimo kot $\langle A_l, A_t, A_M, A_T \rangle$, kjer je A_T seznam pojavitvenih časov dogodkov aktivacije, definiran tako:

$$A_T(P_1^0) = [A_t(P_1^0)] \tag{3.15}$$

$$A_T(P_i^n) = [A_T(P_k 0^n - 1), A_T(P_{k_j}^n - 1)] \tag{3.16}$$

Magnitude nato izračunamo na sledeč način:

$$A_M(P_1^0) = 1 \tag{3.17}$$

$$A_M(P_i^n) = w_j(\frac{2^{n+1} - N_{gap}}{2^{n+1}} - \delta * N_{gap}) \tag{3.18}$$

$$w_j = \begin{cases} 1: & \delta_{Tj} < \tau_W \\ 0: & \delta_{Tj} \geq \tau_W \end{cases} \tag{3.19}$$

$$\delta_{Tj} = A_t(P_{k_j}^{n-1}) - A_t(P_{k_0}^{n-1}) \tag{3.20}$$

$$\tau_W = \min(2^{(n+1)}, t_l) \tag{3.21}$$

kjer δ predstavlja parameter, ki določa jakost vplivanja števila razmakov v vzorcu na magnitudo aktivacije in je v našem primeru nastavljen na vrednost $0,1$. N_{gap} pa je število razmakov večjih od 1, ki ga za $A_m(P_i^n)$ na podlagi naraščajoče urejenega seznama pojavitvenih časov $A_T(P_i^n)$ lahko definiramo tako:

$$N_{gap} = \sum_{q=0}^{l-1} g(A_T(P_i^n), q)) \tag{3.22}$$

$$g(x, q) = \lfloor x[q+1] - x[q] \rfloor \tag{3.23}$$

Iskanje pomembnih vzorcev Pri analizi rezultatov prilagojenega modela smo ugotovili, da se večglasni vzorci zelo pogosto pojavljajo. Zato smo pri izbiri najpomembnejših vzorcev dodali izbor na podlagi mere pokritja, ki ga je definiral Meredith [4]. Za vsak vzorec izračunamo mero pokritja, ki določa, koliko skladbe vzorec z vsemi svojimi pojavitvami pokrije, in jo izračunamo na sledeč način:

$$C(\mathcal{S}, P_i^n) = \bigcup_{k=1}^{k=a_i} A_{Tk}(P_i^n), \tag{3.24}$$

kjer je a_i število aktivacij dela P_i^n, A_{Tk} pa je A_T $k-te$ aktivacije dela P_i^n.

$$COV(P_i^n) = \frac{|C(\mathcal{S}, P_i^n)|}{|\mathcal{S}|}, \tag{3.25}$$

, kjer $COV(P)$ predstavlja mero pokritja, ki jo računamo. Med najdenimi vzorci ohranimo τ_N vzorcev z največjo mero pokritja, τ_N je prag, ki določa število vzorcev, ki jih želimo ohraniti.

Poglavje 4

Vizualizacija

Za analizo in interpretacijo rezultatov smo implementirali spletno aplikacijo, kjer lahko pregledujemo najdene vzorce, ekspertne anotacije in ocene različnih verzij rezultatov. Osnovni pogled aplikacije je predstavljen na Sliki 4.1.

4.1 Uporabljene tehnologije

Za razvoj upodobitve smo uporabili spletne tehnologije, ker nam omogočajo največjo dostopnost in prenosljivost med različnimi platformami. Upodobitev smo razvili v Javascriptovem aplikacijskem ogrodju Aurelia. Aurelia nam je omogočila izgradnjo dinamične spletne aplikacije, ki je enostranska *(ang. single-page application)*. Za risanje smo uporabili knjižnico D3, za osnovno oblikovanje strani pa ogrodje Bootstrap.

4.2 Vsebina aplikacije

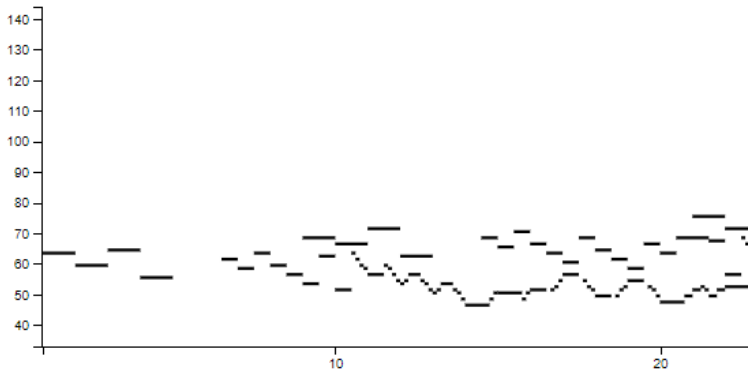
V aplikaciji so prikazani rezultati analize skladbe z naučenim modelom, ekspertna anotacija in celotna skladba. Prikazujemo jih na grafih, kjer abscisna os predstavlja čas, merjen v udarcih četrtink, ordinatna os pa predstavlja MIDI vrednosti tonskih višin. Za vsak ton, ki se pojavi v skladbi ali v vzorcu,

Slika 4.1: Spletna aplikacija za pregledovanje rezultatov.

je narisan pravokotnik, katerega dolžina je odvisna od trajanja tona. Primer vizualizacije skladbe vidimo na sliki 4.2. V vizualizaciji je skladba predstavljena s tremi grafi, ki so postavljeni tako, da so abscisne osi poravnane in jih lahko primerjamo (slika 4.1). Spodnji graf predstavlja celotno skladbo in ima prikazane vse tone, ki se pojavijo v MIDI zapisu izbrane skladbe. Srednji graf prikazuje vzorce, ki so za posamezno skladbo zapisani v ekspertnih anotacijah, zgornji graf pa predstavlja vzorce, ki smo jih odkrili z modelom.

V zgornjem delu upodobitve je seznam rezultatov z ustrezajočimi izmerjenimi merami uspešnosti, po katerih lahko rezultate sortiramo. S klikom na rezultat se ta prikaže v zgornjem grafu, kjer prikazujemo rezultate.

V skladbi je ponavadi več vzorcev, vsak vzorec ima vedno vsaj še eno ponovitev. V prvem in drugem grafu, kjer prikazujemo ponavljajoče vzorce, smo zato vzorce različno obarvali. Pod rezultati imamo tudi gumbe z ujemajočimi barvami. Če kliknemo na gumb za določen vzorec, se nam na grafu prikaže samo ta vzorec, pod gumbom pa se nam pokažejo gumbi za izbiro



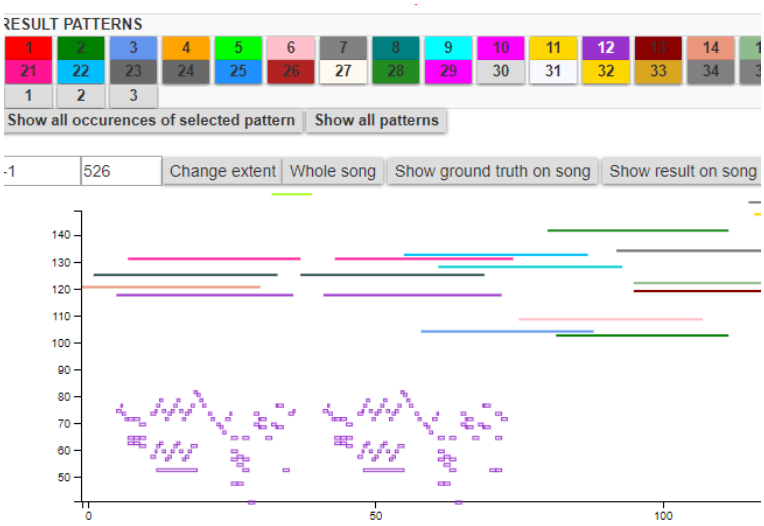
Slika 4.2: Prikaz upodobitve začetka Bachove skladbe. Pravokotniki predstavljajo tone, ki se pojavijo v skladbi.

posameznih pojavitev vzorca, tako da lahko posebej pregledujemo tudi te. Izbira vzorcev in pojavitev je možna tako za zgornji kot za srednji graf. Primer izbire vzorca vidimo na sliki 4.3.

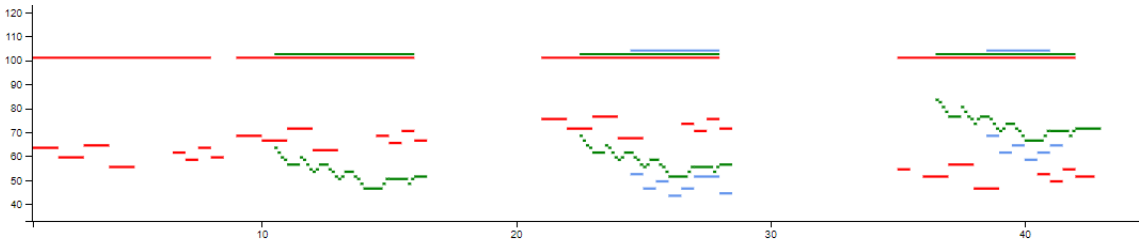
Poleg samih vzorcev na vrhu zgornjega in srednjega grafa vidimo tudi barvne črte, primer vidimo na sliki 4.4. Te predstavljajo časovne intervale, kjer se določen vzorec ponavlja. Za določene vidike analize je tak pogled na vzorce bolj primeren.

Pod seznamom vzorcev imamo možnost izbire prikazanega dela skladbe po času. Če v vnosni polji vnesemo številki začetnega in končnega časa glede na abscisno os, lahko na vseh treh grafih spremenimo obseg vrednosti na abscisni osi. Na ta način lahko bolj natančno pregledujemo določene dele skladbe.

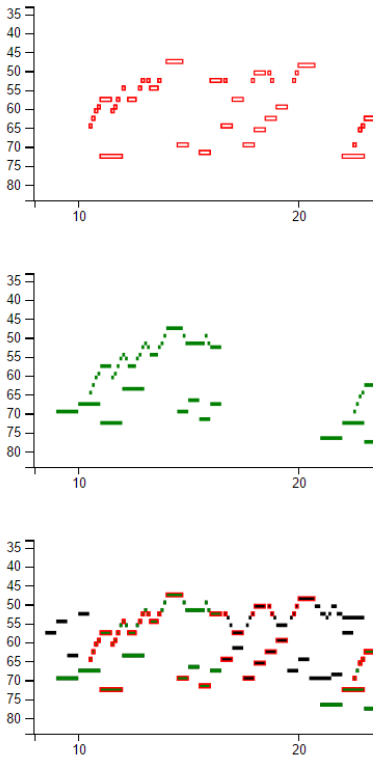
Gumba "Show/Hide ground truth on song" *(slo. Prikaži/Skrij ekspertno anotacijo na skladbi)* in "Show/Hide result on song" *(slo. Prikaži/Skrij rezultat na skladbi)* vklapljata in izklapljata prikaz rezultatov ali ekspertnih anotacij na upodobitvi celotne skladbe, kar je prikazano na sliki 4.5 in nam dobro služi predvsem za primerjavo najdenih vzorcev z anotacijami.



Slika 4.3: Prikaz izbire vzorca. Na sliki smo izbrali vzorec 12. V sivi barvi so prikazani 1, 2 in 3, kar pomeni, da ima vzorec tri ponovitve. Na sliki sta vidni dve, ker gledamo samo začetek skladbe. Če kliknemo na gumbe za pojavitve, se prikaže posamezna pojavitev izbranega vzorca.



Slika 4.4: Na sliki vidimo črte, ki predstavljajo časovne intervale vzorcev.



Slika 4.5: Na sliki vidimo preslikavo prvih dveh grafov na skladbo.

4.2.1 Uporabnost

Aplikacija nam je bistveno pomagala pri analizi rezultatov, analizi napak in pri testiranju pravilnosti naše kode. Ker je pri objektivnih merah, ki so predstavljene v poglavju 5, prisotnih nekaj problemov, nam upodobitev daje tudi dodatno potrditev in oceno delovanja algoritma. Prav tako bi jo lahko v prihodnosti uporabili za primerjavo rezultatov z ostalimi metodami, saj upodablja vzorce v obliki, kot so definirani za oddajo na tekmovanju MIREX.

Poglavje 5

Evalvacija

5.1 Vhodni podatki

Iskanje ponavljajočih vzorcev smo preizkusili na podatkovni zbirki JKUPDD (Johannes Kepler University Development Database). Zbirka je majhna, saj vsebuje le 5 klasičnih skladb z anotiranimi ponavljajočimi vzorci, vendar je trenutno edina javno dostopna zbirka za evalvacijo tovrsnih algoritmov.

Kot osnovo za ponavljajoče vzorce so avtorji zbirke vzeli teme in motive iz:

- Barlow and Morgenstern's (1953) Dictionary of Musical Themes
- Schoenberg's (1967) Fundamentals of Musical Composition
- Bruhn's (1993) J. S. Bach's Well-Tempered Clavier: In-depth Analysis and Interpretation

Za eno od del so anotacijo naredili sami. Pri podajanju podatkovne zbirke se nahaja opozorilo, da anotacije niso popolne, saj sama naloga anotiranja ni dobro definirana.

5.2 Mere za računanje uspešnosti

Za računanje uspešnosti delovanja modela na opravilu iskanja ponavljajočih vzorcev v večglasni glasbi smo poleg standardnih mer (natančnost, priklic in mera F1) uporabili tudi sestavljene mere, ki jih za to opravilo definirata Collins [8] in Meredith [4]. Sestavljene mere so bile uvedene, ker so standardne mere zelo stroge - samo ena napačna nota v vsakem vzorcu pomeni napako. Najdeni vzorci so tako lahko zelo blizu anotiranim, pa so vrednosti standardnih mer vseeno zelo nizke, ker so nerobustne na drobne razlike med najdenimi in iskanimi vzorci, kar pa pri našem opravilu nima smisla, ker ena nota manj ali več v vzorcu ne predstavlja bistvene razlike pri kakovosti rezultata. Poleg standardnih mer zato uporabimo še vzpostavitvene mere, pojavitvene mere in tronivojske mere. Za predstavitev mer bomo v nadaljevanju uporabljali naslednje oznake:

- $n_{\mathcal{P}}$ - število anotiranih vzorcev
- $\Pi = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_{n_{\mathcal{P}}}\}$ - množica anotiranih vzorcev v skladbi *(ang. ground truth)*
- $\mathcal{P} = \{P_1, P_2, \ldots, P_{m_P}\}$ - anotirane pojavitve anotiranega vzorca $\mathcal{P}$
- $n_{\mathcal{Q}}$ - število vzorcev v množici, ki jo vrne algoritem
- $\Xi = \{\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_{n_{\mathcal{Q}}}\}$ - množica vzorcev, ki jih odkrije algoritem
- $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_{m_Q}\}$ - pojavitve odkritega vzorca $\mathcal{Q}$.
- k - število vzorcev, ki jih najde algoritem in se ujemajo z anotiranimi vzorci

Standardne mere

Standardna natančnost *(ang. precision)*:

$$P = k/n_Q \tag{5.1}$$

Standardni priklic *(ang. recall)*:

$$R = k/n_P \tag{5.2}$$

Standardna mera F1 *(ang. F1 score)*:

$$F1 = 2 * PR/(P + R) \tag{5.3}$$

Vzpostavitvene mere Vzpostavitvene mere visoko ocenijo rezultate, ki so našli vsaj eno pojavitev vsakega vzorca. Za izračun vzpostavitvenih mer najprej izračunamo podobnost simbolične glasbe za vse kombinacije pojavitev v dveh vzorcih $\mathcal{P}$ in $\mathcal{Q}$.

Podobnost simbolične glasbe za pojavitvi P_i in Q_j:

$$s(P_i, Q_j) : |P_i \cap Q_j| / \max\{|P_i|, |Q_j|\} \tag{5.4}$$

Vse kombinacije pojavitev dveh vzorcev zapišemo v matriko rezultatov:

$$s(\mathcal{P}, \mathcal{Q}) = \begin{bmatrix} s(P_1, Q_1) & s(P_1, Q_2) & \dots & s(P_1, Q_{m_Q}) \\ s(P_2, Q_1) & s(P_2, Q_2) & \dots & s(P_2, Q_{m_Q}) \\ \vdots & \vdots & \ddots & \vdots \\ s(P_{m_P}, Q_1) & s(P_{m_P}, Q_2) & \dots & s(P_{m_P}, Q_{m_Q}) \end{bmatrix} \tag{5.5}$$

Iz matrike rezultatov zgradimo vzpostavitveno matriko. Uporabimo $S(\mathcal{P}, \mathcal{Q})$, kar predstavlja maksimalno vrednost v matriki rezultatov $s(\mathcal{P}, \mathcal{Q})$:

$$S(\Pi, \Xi) = \begin{bmatrix} S(\mathcal{P}_1, \mathcal{Q}_1) & S(\mathcal{P}_1, \mathcal{Q}_2) & \dots & S(\mathcal{P}_1, \mathcal{Q}_{n_\mathcal{Q}}) \\ S(\mathcal{P}_2, \mathcal{Q}_1) & S(\mathcal{P}_2, \mathcal{Q}_2) & \dots & S(\mathcal{P}_2, \mathcal{Q}_{n_\mathcal{Q}}) \\ \vdots & \vdots & \ddots & \vdots \\ S(\mathcal{P}_{n_\mathcal{P}}, \mathcal{Q}_1) & S(\mathcal{P}_{n_\mathcal{P}}, \mathcal{Q}_2) & \dots & S(\mathcal{P}_{n_\mathcal{P}}, \mathcal{Q}_{n_\mathcal{Q}}) \end{bmatrix} \tag{5.6}$$

Iz tega izračunamo vzpostavitvene mere:
Vzpostavitvena natančnost *(ang. establishment precision)*:

$$P_{est} = \frac{1}{n_\mathcal{Q}} \sum_{j=1}^{n_\mathcal{Q}} \max\{S(\mathcal{P}_i, \mathcal{Q}_j) | i = 1 \dots n_\mathcal{P}\} \tag{5.7}$$

Vzpostavitveni priklic *(ang. establishment recall)*:

$$R_{est} = \frac{1}{n_{\mathcal{P}}} \sum_{j=1}^{n_{\mathcal{P}}} \max\{S(\mathcal{P}_i, \mathcal{Q}_j) | i = 1 \dots n_{\mathcal{Q}}\} \tag{5.8}$$

Vzpostavitvena mera F1 *(ang. establishment F1 score)*:

$$F1_{est} = 2 * P_{est} R_{est} / (P_{est} + R_{est}) \tag{5.9}$$

Pojavitvene mere Pojavitvena mera visoko oceni rezultate, ki najdejo vse pojavitve najdenega vzorca, čeprav kakšnega vzorca sploh ni med rezultati. Za izračun definiramo pojavitveno matriko $O(\Pi, \Xi)$. Uvedemo prag c, ki določa vrednost podobnosti simbolične glasbe med dvema pojavitvama, ki mora biti dosežena, da za najdeno pojavitev potrdimo, da se je pojavila v ekspertni anotaciji. Uporabili bomo meri, kjer je $c = 0.5$ in $c = 0.75$. Najprej v množico $\mathcal{I}$ dodamo vse indekse vzpostavitvene matrike, kjer je vrednost večja od praga c. Potem naredimo pojavitveno matriko velikosti $n_{\mathcal{P}} \times n_{\mathcal{Q}}$, kjer so vsi elementi 0. Zapolnimo jo na sledeči način:

$$\forall (i, j) \in \mathcal{I} : O(\Pi, \Xi)[i, j] = s(\mathcal{P}_i, \mathcal{Q}_j) \tag{5.10}$$

Iz tega izračunamo pojavitvene mere:
Pojavitvena natančnost *(ang. occurrence precision)*:

$$P_{occ} = \frac{1}{n_{col}} \sum_{j=1}^{n_{\mathcal{Q}}} O(i, j) | i = 1 \dots n_{\mathcal{P}}, \tag{5.11}$$

kjer n_{col} predstavlja število neničelnih stolpcev v matriki O.
Pojavitveni priklic *(ang. occurrence recall)*:

$$R_{occ} = \frac{1}{n_{row}} \sum_{j=1}^{n_{\mathcal{P}}} S(i, j) | i = 1 \dots n_{\mathcal{Q}}, \tag{5.12}$$

kjer n_{row} predstavlja število neničelnih vrstic v matriki O.
Pojavitvena mera F1 *(ang. occurence F1 score)*:

$$F1_{occ} = 2 * P_{occ} R_{occ} / (P_{occ} + R_{occ}) \tag{5.13}$$

Tronivojske mere Tronivojska mera je kompromis med pojavtivenimi in vzpostavitvenimi merami. Imenuje se tronivojska, ker meri pravilnost na treh različnih strukturnih nivojih: med množicama Π in Ξ, med vsemi množicami $\mathcal{P}$ in $\mathcal{Q}$ in med vsemi vzorci Q in P. Za pridobitev tronivojskih mer najprej definiramo naslednje mere:

$$P_1(P, Q) = |P \cap Q|/|Q| \tag{5.14}$$

$$R_1(P, Q) = |P \cap Q|/|P| \tag{5.15}$$

$$F_1(P, Q) = \frac{2P_1(P, Q)R_1(P, Q)}{P_1(P, Q) + R_1(P, Q)} \tag{5.16}$$

$$P_2(\mathcal{P}, \mathcal{Q}) = \frac{1}{m_Q} \sum_{l=1}^{m_Q} \max\{F_1(P_k, Q_l)|k = 1 \dots m_P\} \tag{5.17}$$

$$R_2(\mathcal{P}, \mathcal{Q}) = \frac{1}{m_P} \sum_{k=1}^{m_P} \max\{F_1(P_k, Q_l)|l = 1 \dots m_Q\} \tag{5.18}$$

$$F_2(\mathcal{P}, \mathcal{Q}) = \frac{2P_2(\mathcal{P}, \mathcal{Q})R_2(\mathcal{P}, \mathcal{Q})}{P_2(\mathcal{P}, \mathcal{Q}) + R_2(\mathcal{P}, \mathcal{Q})} \tag{5.19}$$

Iz teh mer dobimo tronivojske mere:
Tronivojska natančnost *(ang. three-layer precision)*:

$$P_3(\Pi, \Xi) = \frac{1}{n_\mathcal{Q}} \sum_{j=1}^{n_\mathcal{Q}} \max\{F_2(\mathcal{P}_i, \mathcal{Q}_j)|i = 1 \dots n_\mathcal{P}\} \tag{5.20}$$

Tronivojski priklic *(ang. three-layer recall)*:

$$R_3(\Pi, \Xi) = \frac{1}{n_\mathcal{P}} \sum_{i=1}^{n_\mathcal{P}} \max\{F_2(\mathcal{P}_i, \mathcal{Q}_j)|j = 1 \dots n_\mathcal{Q}\} \tag{5.21}$$

Tronivojska mera F1 *(ang. three-layer F1 score) (v nadaljevanju TLF)*:

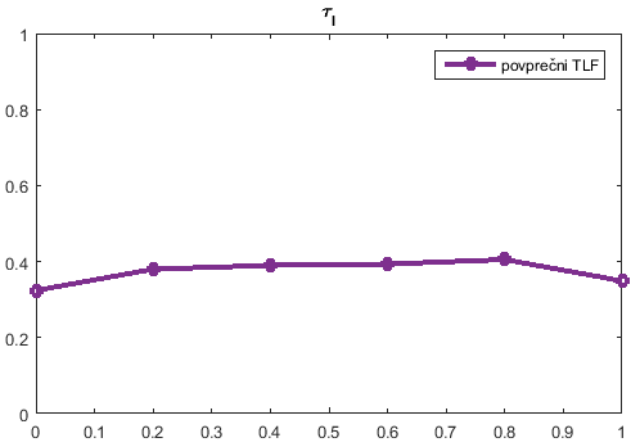
$$TLF(\Pi, \Xi) = \frac{2P_3(\Pi, \Xi)R_3(\Pi, \Xi)}{P_3(\Pi, \Xi) + R_3(\Pi, \Xi)} \tag{5.22}$$

Parametri Model lahko prilagajamo z velikim številom parametrov. Na podlagi preteklih del, ki so uporabljala CHM za reševanje problemov, smo določili vrednosti parametrov:

$$[\tau_I;\ \tau_H;\ \omega;\ \phi;\ \gamma] = [0.9;\ 0.2;\ 2.5;\ 0.00005;\ 0.05] \tag{5.23}$$

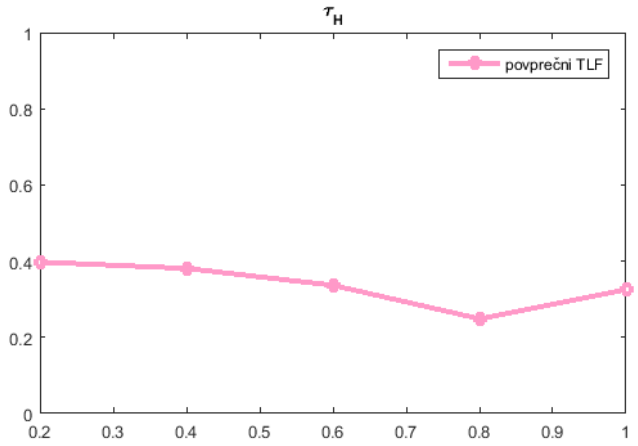
Na podlagi rezultatov modela na zbirki JKUPDD smo naredili občutljivostno analizo modela na parametre. Ugotovili smo, da je rezultat na spremembe precej odporen. Za vrednosti smo ohranili začetne vrednosti, ki so se tudi na zbirki JKUPDD dobro izkazale. Parametri, ki smo jih opazovali, so našteti spodaj, grafi poleg parametrov pa za bazo JKUPDD prikazujejo občutljivost mere TLF, ki je definirana v 5.2, na spreminjanje vrednosti tega parametra. Abscisna os na grafih predstavlja vrednosti parametra, ordinatna pa mero TLF:

- τ_I - uravnavanje jakosti mehanizma inhibicije.



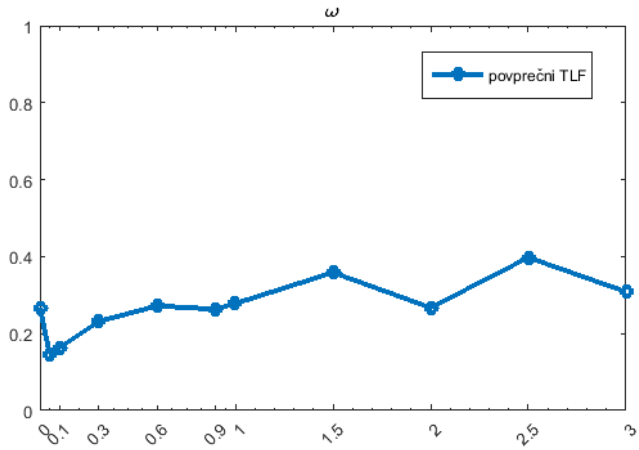
Slika 5.1: Graf prikazuje občutljivost mere TLF na spreminjanje parametra τ_L, na zbirki JKUPDD.

- τ_H - uravnavanje jakosti mehanizma halucinacije.



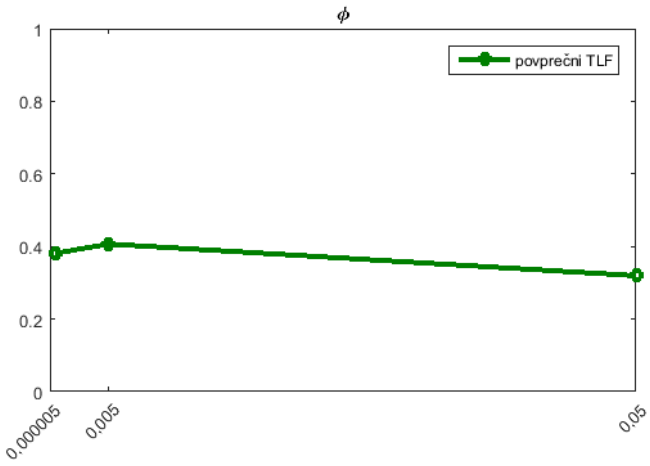
Slika 5.2: Graf prikazuje občutljivost mere TLF na spreminjanje parametra τ_H na zbirki JKUPDD.

- ω - parameter se uporablja pri izbiri delov pri učenju hierarhije. Določa prag pokritja, ki mora biti dosežen, da nehamo izbirati dele.



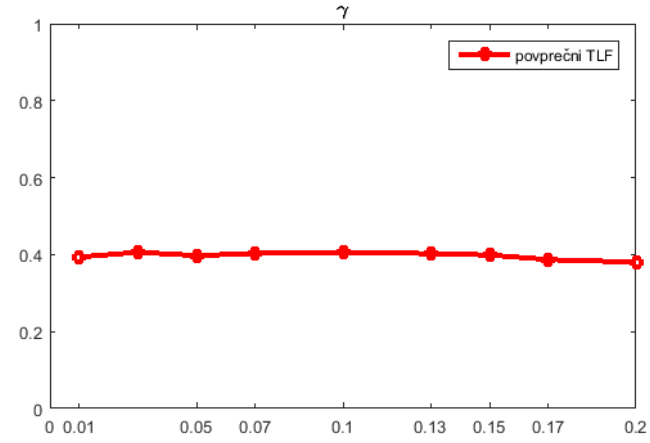
Slika 5.3: Graf prikazuje občutljivost mere TLF na spreminjanje parametra ω, na zbirki JKUPDD.

- ϕ - parameter se uporablja pri izbiri delov pri učenju hierahije. Določa minimalen doprinos k pokritju, ki ga mora del dodati, da se izbiranje delov še naprej vrši.



Slika 5.4: Graf prikazuje občutljivost mere TLF na spreminjanje parametra ϕ na zbirki JKUPDD.

- γ - pri učenju hierarhije določa prag vrednosti v histogramu sopojavitev dveh tonov, da nek del dodamo med kandidate za dele.



Slika 5.5: Graf prikazuje občutljivost mere TLF na spreminjanje parametra ω na zbirki JKUPDD.

5.3 Rezultati

Rezultati naše rešitve na bazi JKUPDD so za vsako pesem posebej prikazani v tabeli 5.1. Vidimo, da so pojavitvene mere pri $c = 0.75$ večinoma enake nič, kar pomeni, da je problem pri iskanju pojavitev vzorca, če pri tem želimo

Tabela 5.1: Rezultati za posamezno skladbo iz baze JKUPDD, ki smo jih dosegli z našim modelom.

Ime skladbe	$P_{occ(c=.75)}$	$R_{occ(c=.75)}$	$F_{1occ(c=.75)}$	$P_{occ(c=.5)}$	$R_{occ(c=.5)}$	$F_{1occ(c=.5)}$
beethovenOp2No1Mvt3	0	0	0	0.64	0.64	0.64
bachBWV889Fg	0	0	0	0	0	0
chopinOp24No4	0	0	0	0.53	0.57	0.55
gibbonsSilverSwan1612	0	0	0	0.60	0.70	0.65
mozartK282Mvt2	0.84	0.84	0.84	0.68	0.77	0.72
	P_{est}	R_{est}	$F1_{est}$	P_3	R_3	TLF_1
beethovenOp2No1Mvt3	0.44	0.47	0.46	0.50	0.51	0.51
bachBWV889Fg	0.22	0.17	0.19	0.16	0.14	0.15
chopinOp24No4	0.35	0.39	0.37	0.35	0.38	0.37
gibbonsSilverSwan1612	0.60	0.22	0.33	0.58	0.25	0.35
mozartK282Mvt2	0.63	0.55	0.59	0.71	0.57	0.63

75-odstotno ujemanje dogodkov v pojavitvi ekspertne anotacije in našega rezultata, da potrdimo, da se je ta pojavila. Mere se pri $c = 0.5$ izboljšajo, kar pomeni, da najdemo pojavitve vzorcev, a vključimo zraven preveč ali premalo dogodkov. Vidimo, da po merah izstopa Bachova skladba, ki ima nižje mere od ostalih. Po pregledu rezultatov smo ugotovili, da se anotirani ekspertni vzorci pri Bachovi skladbi razlikujejo po tem, da so kratki in enoglasni, naš model pa vrne daljše vzorce z več dogodki.

5.4 Primerjava z ostalimi

Rezultate modela lahko primerjamo z rezultati ostalih raziskovalcev, ki so na tekmovanju MIREX reševali enak problem *(repeated pattern discovery symPoly)* na enaki podatkovni zbirki in z istimi merami. Opravilo se je na tekmovanju MIREX začelo izvajati leta 2013. Leta 2015 ni bilo oddane nobene rešitve za ta problem, zadnji dostopni rezultati pa so iz leta 2016.

Tabela 5.2: Tabela prikazuje imena algoritmov z njihovimi avtorji in letnico oddaje na tekmovanju MIREX.

Ime	Avtor	Leto oddaje
SIATECCompress-TLF1	David Meredith	2016
MotivesExtractor	Oriol Nieto, Morwaread Farbood	2014
Motives_poly	Oriol Nieto, Morwaread Farbood	2013
SIATECCompressSegment	David Meredith	2013
SIATECCompressRaw	David Meredith	2013
SIATECCompressBB	David Meredith	2013
COSIATECSegment	David Meredith	2013
COSIATECRaw	David Meredith	2013
COSSIATECBB	David Meredith	2013

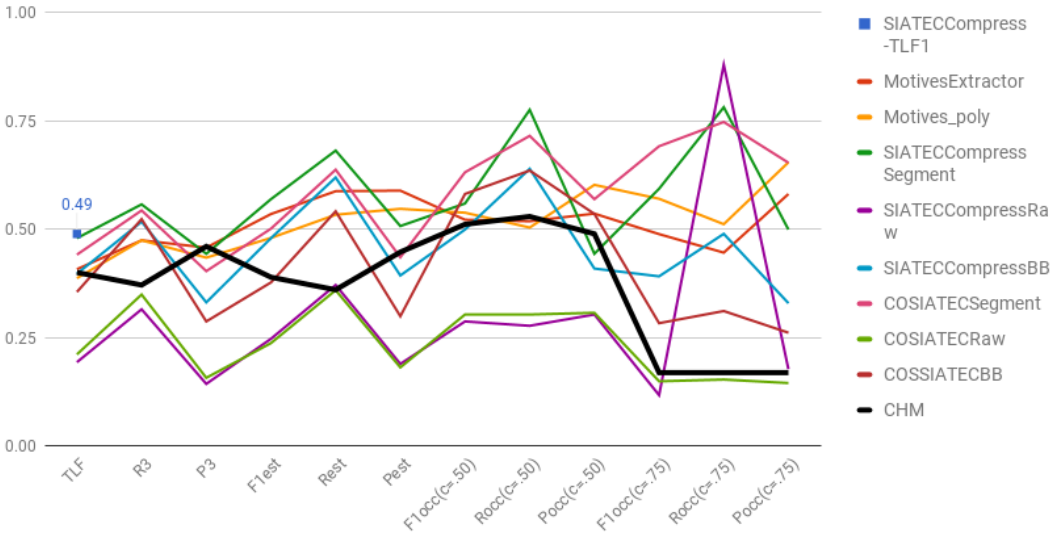
Vsega skupaj je bilo na tem opravilu preizkušenih 11 algoritmov treh različnih avtorjev, seznam algoritmov je v tabeli 5.2. Algoritmi z enakimi avtorji se zadeve večinoma lotevajo po enakem principu, z izboljšavami prejšnjih verzij.

Leta 2016 je Meredith oddal 3 algoritme. Povsod gre z algoritem SIATECCOMPRESS, le da vsaka oddaja različno nastavi parametre, tako da ena verzija maksimizira tronivojsko natančnost, druga tronivojski priklic, tretja pa tronivojsko mero F1. Za primerjavo bomo uporabili le slednjega, ker je najbolj relevanten. V tabeli 5.3 so rezultati za vse obstoječe algoritme. Rezultati so povprečje mer za vse skladbe baze JKUPDD.

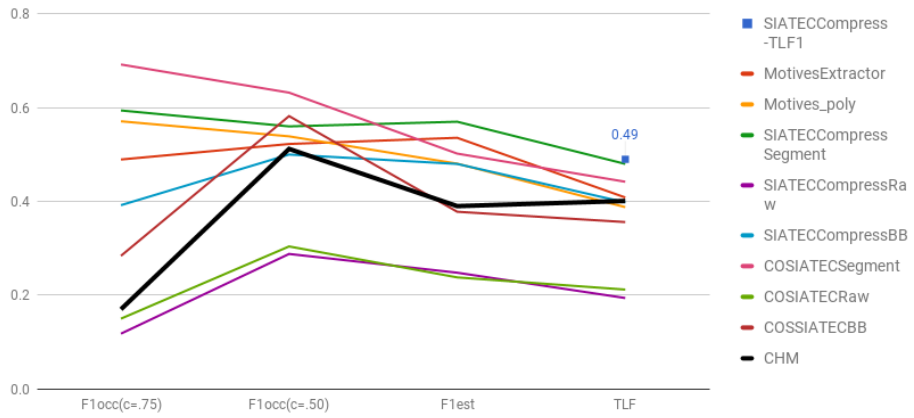
Za lažjo primerjavo so rezultati prikazani tudi grafično. Na sliki 5.6 so prikazane vse mere za vse algoritme, na sliki 5.7 pa samo F mere. Naši rezultati so pod oznako CHM.

Tabela 5.3: Rezultati za posamezno skladbo iz baze JKUPDD, ki smo jih dosegli z našim modelom.

Ime algoritma	$P_{occ(c=.75)}$	$R_{occ(c=.75)}$	$F_{1occ(c=.75)}$	$P_{occ(c=.5)}$	$R_{occ(c=.5)}$	$F_{1occ(c=.5)}$
SIATECCompress-TLF1	x	x	x	x	x	x
MotivesExtractor	0.5819	0.4467	0.4894	0.5367	0.5191	0.5255
Motives_poly	0.6550	0.5123	0.5709	0.6030	0.5045	0.5338
SIATECCompressSegment	0.5	0.782	0.594	0.444	0.776	0.56
SIATECCompressRaw	0.178	0.088	0.118	0.304	0.278	0.288
SIATECCompressBB	0.33	0.49	0.392	0.41	0.64	0.5
COSIATECSegment	0.654	0.748	0.692	0.57	0.716	0.632
COSIATECRaw	0.146	0.154	0.15	0.308	0.304	0.304
COSSIATECBB	0.262	0.312	0.284	0.536	0.636	0.582
CHM	0.17	0.17	0.17	0.49	0.53	0.512
	P_{est}	R_{est}	$F1_{est}$	P_3	R_3	TLF_1
SIATECCompress-TLF1	x	x	x	x	x	0.4897
MotivesExtractor	0.5897	0.5883	0.5357	0.4583	0.4751	0.4084
Motives_poly	0.5477	0.5341	0.4806	0.4351	0.4751	0.3878
SIATECCompressSegment	0.508	0.682	0.57	0.444	0.558	0.48
SIATECCompressRaw	0.19	0.372	0.248	0.144	0.316	0.194
SIATECCompressBB	0.394	0.62	0.48	0.332	0.518	0.398
COSIATECSegment	0.436	0.638	0.502	0.404	0.544	0.442
COSIATECRaw	0.182	0.36	0.238	0.158	0.35	0.212
COSSIATECBB	0.3	0.542	0.378	0.288	0.524	0.356
CHM	0.448	0.361	0.39	0.461	0.372	0.401



Slika 5.6: Na sliki vidimo graf za pojavitvene mere, vzpostavitvene mere in tronivojske mere za vse obstoječe algoritme, ki so bili preizkušeni na bazi JKUPDD. Naš algoritem smo označili s črno barvo.



Slika 5.7: Na sliki vidimo graf za pojavitveno mero F1, ko je $c = 0.75$ in $c = 0.50$, vzpostavitveno mero F1 in tronivojsko mero F1 za vse obstoječe algoritme, ki so bili preizkušeni na bazi JKUPDD. Naš algoritem smo označili s črno barvo.

5.5 Diskusija

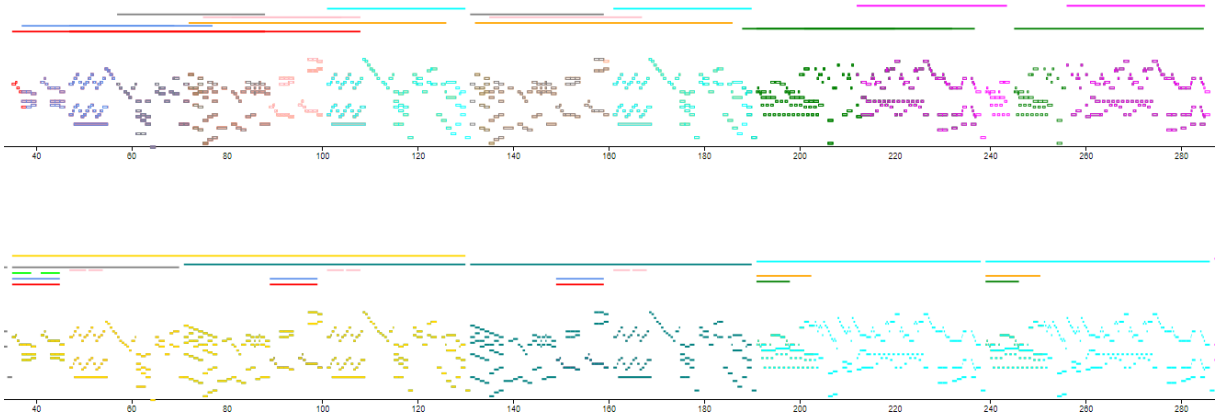
Rezultati predlaganega modela so, glede na tronivojsko mero F1, nekje v sredini med primerjanimi algoritmi - trije algoritmi so boljši, trije pa slabši. Prvo, kar lahko opazimo, je, da se pojavitvena mera pri našem pristopu zelo razlikuje, ko znižamo prag c. Iz tega lahko sklepamo, da algoritem najde okvirne vzorce, vendar je problem v točnem ujemanju, kar smo opazili tudi v upodobitvi in opisali v podpoglavju 5.5.2. Pojavitvena mera F1 je pri $c = 0.50$ večja kot vzpostavitvena in glede na to, da vzpostavitvena mera ni večja od 0.5, lahko rečemo, da model bolje išče pojavitve posameznega vzorca kot pa odkriva prave pomembne vzorce. Nekaterih vzorcev ne odkrijemo, ker kljub mehanizmom, ki jih model vsebuje, manjkajo mehanizmi za zaznavanje analitično pomembnega vzorca.

Navkljub ne najboljšim kvantitativnim rezultatom je iz vizualizacije najdenih vzorcev razvidno, da so rezultati kompozicionalnega hierarhičnega modela kljub vsemu zanimivi za analizo, saj so najdeni vzorci večinoma smiselni, kar nas lahko napelje tudi na težave z ekspertnimi anotacijami zbirke JKUPDD. Primer rezultatov na Mozartovi skladbi, kjer naš model najde 13 zanimivih vzorcev, ekspertna anotacija pa jih določa 11, je prikazan na sliki 5.8.

5.5.1 Problem zbirke JKUPDD

Zbirka JKUPDD ima dva glavna problema, zaradi katerih se pri razvoju modela nismo preveč posvečali številkam, ki jih dajo mere, ampak nam je ta služila le za neko osnovno primerjavo z ostalimi in pa za analizo napak, ki smo jih sproti odkrivali s pomočjo naše vizualizacije in transparentnosti modela.

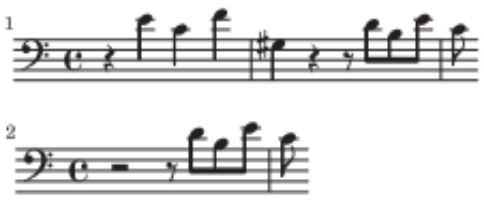
Verodostojnost David Meredith v članku [4] v poglavju 4.2.4 na široko opiše problem anotacije JKUPDD. Opozori na vprašanje, do kakšne mere so rezultati v anotaciji JKUPDD pravilni in na vprašanje ali JKUPDD sploh



Slika 5.8: Na sliki vidimo vizualizacijo vzorcev v delu Mozartove skladbi. Zgornji graf prikazuje rezultate, ki jih je vrnil naš algoritem, spodnji graf pa prikazuje ekspertno anotacijo pomembnih ponavljajočih vzorcev. Vidimo, da so najdeni vzorci pomembna ponavljanja ali del njih. Opazimo pa problem določanja pomembnosti vzorcev. Znotraj temno zelenega vzorca v rezultatih model odkrije ravno drug del tega vzorca, kot pomemben vzorec. Najden poddvzorec znotraj temno zelenega vzorca vidimo v roza barvi, na spodnjem grafu pa pravilna podvzorca v rumeni in temno zeleni barvi. Vidimo tudi, da namesto nekaterih daljših anotiranih vzorcev najdemo več krajših. Tak primer sta siv, oranžen in temno moder vzorec v naših rezultatih, ki bi skupaj lahko predstavljala svetlo rumen anotiran vzorec.

zagotavlja pravilne odgovore za problem odkrivanja ponavljajočih tem in vzorcev. Medtem, ko naš algoritem najde v posamezni skladbi več deset ponavljajočih vzorcev, je anotiranih vzorcev v JKUPDD le od 3 do 9 na posamezno skladbo, kar pomeni, da zbirka JKUPDD ne odgovarja na vprašanje kateri vzorci se v dani skladbi ponavljajo, ampak kateri vzorci se v skladbi ponavljajo in so zaznavno in analitično pomembni. Problem nastane pri vprašanju, če obstaja ciljna skupina ljudi, ki se strinja, kateri vzorci so v določeni skladbi pomembni. Če taka skupina ne obstaja, potem se moramo vprašati po smiselnosti avtomatskega generiranja baze JKUPDD [4]. Collins [29] je naredil preizkus, kjer so študenti, ki so imeli posebna usposabljanja za to nalogo, ocenjevali pomembnost vzorcev v Chopinovih delih. Ocenjevali so, kateri deli so dovolj pomembni, da bi jih vključili v esej o analizi skladbe. Odkril je, da lahko s formulo, ki na podlagi števila pojavitev, kompaktnosti in stisljivosti določa pomembnost vzorca, pojasni 70 odstotkov variance odgovorov študentov, po drugi strani pa nobena napoved posameznika ni dosegla 70 odstotkov soglasnih ocen. Po tem lahko sklepamo, da pri zaznavanju pomembnosti vzorcev prihaja do večjih razlik med posamezniki.

Vprašanje je, do kakšne mere je zbirka JKUPDD verodostojna, glede na to, da ni bila pridobljena na podlagi empiričnega eksperimenta, ampak so vzorce določili posamezni muzikologi. Vprašanje je tudi, če so deli, ki so določeni skupini pomembni, res najbolj uporabni za analizo in nadaljnjo uporabo teh vzorcev v objektivno usmerjenih opravilih pridobivanja informacij iz glasbe, kot so prepoznavanje skladateljev, klasifikacija skladb, odkrivanje napak v zapisih itd. Lahko trdimo, da so vzorci v JKUPDD pomembni za analitika, ki jih je zapisal, na podlagi zgoraj omenjenega eksperimenta pa tega ne moremo posplošiti na druge skupine ljudi. Kljub temu ne moremo zanemariti pomembnosti zbirke, ker je bila večina anotiranih vzorcev med muzikologi potrjenih. Za vse vzorce v zbirki JKUPDD to ne velja, ker je avtor zbirke nekaj vzorcev dodal sam. Meredith poda tudi nekaj konkretnih primerov vzorcev, ki jih njegov algoritem najde, pa niso anotirani kot pomembni vzorci, čeprav gre za pojavitve ponovitev dela glavne teme, ki pa je definirana. Primer je



Slika 5.9: Na primeru pri številki 1 vidimo glavno temo, ki jo v Bachovi skladbi definira Bruhn. Pod številko 2 vidimo vzorec, ki se v skladbi večkrat ponovi in je del glavne teme, v anotiranih vzorcih pa ni prisoten.

na sliki 5.9.

Majhnost Največji problem zbirke JKUPDD je ta, da vsebuje le 5 skladb. Vsi avtorji skladb prihajajo iz Evrope. Obdobja skladateljev so različna - renesansa, barok, klasicizem in romantika, a zbirka kljub temu zagotavlja preverjanje delovanja iskanja ponavljajočih vzorcev le na glasbi podobnega kulturnega izvora, ki vsebuje določene skupne lastnosti. Ne moremo trditi, da visoke mere uspešnosti na teh skladbah pomenijo, da bodo algoritmi uspešno delovali na kakršnikoli glasbi. Druga anotirana zbirka za problem ne obstaja, zato je to trenutno edina možnost za primerjavo različnih algoritmov in se pri reševanju istega problema uporablja za referenco na najbolj znanih dogodkih na temo pridobivanja informacij iz glasbe (MIREX, ISMIR).

Kljub majhnosti pa pri učenju modela ne pridemo do prevelikega prilagajanja modela podatkom *(ang. overfitting)*, ker naš model deluje na podlagi nenadzorovanega učenja in parametrov modela nismo prilagajali glede na rezultate zbirke. Model se ne uči na podlagi anotacij posamezne skladbe. Te smo uporabili le za primerjavo z ostalimi in neko povratno informacijo.

Na podlagi zgoraj opisanih problemov moramo pomembnost rezultatov na bazi JKUPDD jemati z rezervo. Bolj relevantno ocenjevanje bi bilo doseganje objektivno merljivih rezultatov na drugih opravilih, ki jih lahko rešujemo na

podlagi pridobljenih ponavljajočih vzorcev.

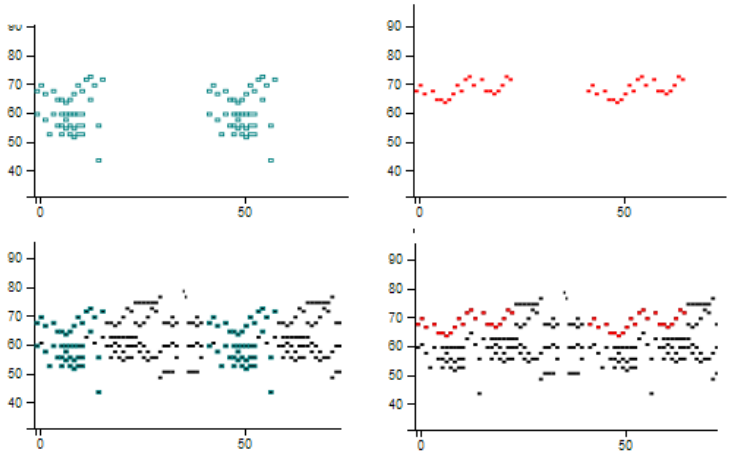
5.5.2 Analiza napak

Kompozicionalni hierarhični model iz skladb pridobi ponavljajoče vzorce. Analiza napak glede na anotirane podatke je pokazala sledeče težave pri odkrivanju vzorcev:

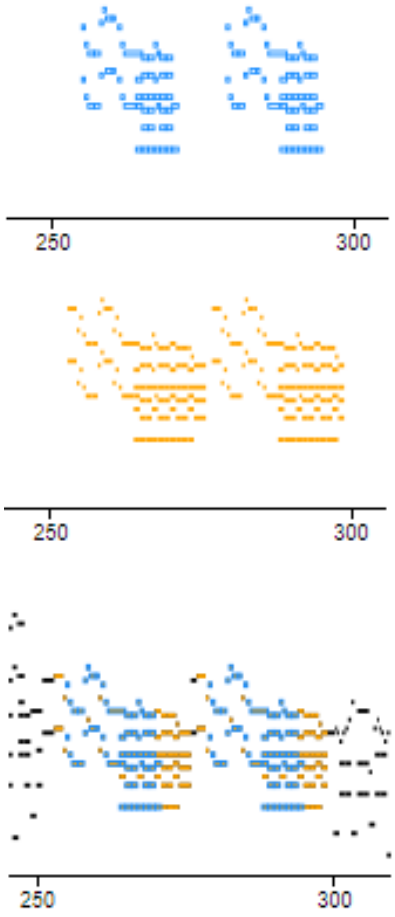
Pomembne enoglasne linije Največji problem modela je odkrivanje pomembnih linij znotraj ponovitve večglasnega segmenta. Naš model kot ponavljajoči vzorec vzame celoten ponavljajoči segment, ker nima mehanizma za zaznavanje pomembne enoglasne linije znotraj tega vzorca. Primer takšne napake je predstavljen na sliki 5.10.

Začetek in konec ponavljajočih vzorcev Drugi najpogostejši problem pri iskanju vzorcev je časovni obseg ponavljajočega vzorca. Ko odkrijemo dve ponavljanji, naš model večkrat napačno definira začetek in konec vzorca v primerjavi z anotiranimi primeri. Primer take napake je na sliki 5.11. Glede na to, da neko jedro in prisotnost istega vzorca vseeno odkrijemo, je to zanemarljivejši problem, kljub vsemu pa bi bilo smiselno poiskati mehanizem, ki bi določal glasbeno pomemben začetek in konec nekega vzorca. Od tu prihajajo nizke pojavitvene mere.

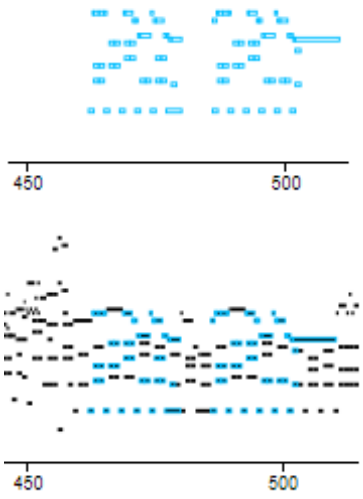
Odkrivanje "nepomembnih"vzorcev Zadnji problem je ta, da naš model odkrije preveč vzorcev. Na koncu jih filtriramo, ampak še vedno je v naši množici veliko vzorcev, ki jih ni v ekspertni anotaciji iz JKUPDD. Primer takega vzorca je prikazan na sliki 5.12. Na podlagi problemov podatkovne zbirke, ki so opisani v 5.5.1, ni nujno, da je to dejanski problem. Je le problem pri doseganju višjih ocen na zbirki JKUPDD, v resnici pa so lahko ti vzorci v kontekstu analize dela vseeno pomembni.



Slika 5.10: Na sliki vidimo začetek Beethovnove skladbe. Vsi štirje grafi predstavljajo isti del iste skladbe. Na ordinatni osi so MIDI vrednosti, na abscisni osi pa je čas. V prvi vrsti na levi vidimo dve ponovitvi vzorca, ki ga najde naš algoritem, na desni pa dve ponovitvi anotiranega vzorca, ki ga naš algoritem ni odkril. V spodnji vrsti sta vzorca preslikana na prikaz vseh not v tem odseku skladbe. Vidimo, da naš algoritem iz dveh ponavljanj ne zna izluščiti pomembne linije, kljub vsemu pa odkrije območje pomembnega vzorca. Kljub temu, da zaradi takih napak dosegamo nižje rezultate na bazi JKUPDD, vzorec, ki ga najdemo, v sebi skriva pravilno informacijo in bi mogoče pri analizi ali drugih opravilih na podlagi vzorcev služil enako dobro kot izluščen vzorec, ki so ga odkrili eksperti.



Slika 5.11: Na sliki vidimo problem neujemanja začetka in konca vzorcev. Z upodobitvijo, ki je opisana v 4, sta prikazani dve pojavitvi vzorca. V zgornjem grafu sta pojavitvi, ki jih odkrije naš model, v sredini je ekspertna anotacija, spodaj pa sta vzorca preslikana na vhodne podatke, kjer naš vzorec prekriva ekspertno anotacijo. Vidimo, da gre v principu za isto jedro vzorca, vendar naš algoritem ne odkrije pravilnega začetka in konca.



Slika 5.12: Na sliki vidimo vzorec z dvema pojavitvama, ki ga v Chopinovi skladbi odkrije naš algoritem, v ekspertni anotaciji pa se ta ne pojavi. Vidimo, da vizualno gre za ponavljanje, vendar mogoče ta vzorec glasbeno in zaznavno pri poslušanju ni pomemben. Kljub vsemu ne smemo izključevati možnosti, da bi za določeno ciljno skupino ljudi ali pa za opravila, ki jih pri pridobivanju iz glasbe lahko vršimo na podlagi najdenih ponavljajočih vzorcev, tak vzorec bil pomemben.

Poglavje 6

Klasifikacija slovenskih ljudskih pesmi na podlagi ponavljajočih vzorcev

Zaradi omejitev zbirke JKUPDD smo se odločili, da metodo za iskanje ponavljajočih vzorcev preizkusimo tudi na drugačni domeni, in sicer za klasifikacijo ljudskih pesmi v variantne tipe. Zaradi ustne narave ljudskega izročila, se ljudske pesmi pri prenosu neprestano spreminjajo - nastajajo pesemske *variante*. Družina pesmi znotraj istega *variantnega tipa* predstavlja enako pesem, ki se je s časom in lokacijo spreminjala, obdržala pa je enako sporočilo in do neke mere tudi melodijo. Z eksperimentom smo torej želeli preizkusiti, kako uspešna je lahko klasifikacija neznane pesmi v nabor *variantnih tipov* na podlagi vzorcev, ki jih izluščimo z našim pristopom.

Eksperiment smo izvedli na rokopisni zbirki OSNP, ki je nastala med letoma 1906 in 1913, vsebuje pa okoli 12.000 zapisov ljudskih pesmi, ki so v digitalizirani obliki zbrani v digitalnem arhivu Etnomuza Glasbenonarodopisnega inštituta ZRC SAZU. Pesmi v zbirki so večglasne in enoglasne, razdeljene so v variantne tipe, kjer nekateri vsebujejo le po eno pesem, nekateri pa več deset.

6.1 Metoda

Na zbirki smo izvedli tri eksperimente:

1. Izmed variantnih tipov, ki vsebujejo največ pesmi, smo jih izbrali 9, znotraj posameznega variantnega tipa pa smo izbrali največ 20 pesmi. Te pesmi so enoglasne in večglasne.

2. Med vsemi variantnimi tipi smo izbrali 10 tistih, ki imajo največ večglasnih pesmi. Za vsak izbran variantni tip smo izbrali 10 večglasnih pesmi.

3. Vzeli smo izbor pesmi iz prvega eksperimenta in vse večglasne pesmi v izboru transformirali. Na podlagi tega, da človek kot glavno melodijo skladbe največkrat zazna najvišji ton v večglasju in posledično večglasja v slovenski ljudski glasbi večinoma nastajajo z dodajanjem nižjih glasov, smo večglasja spremenili v enoglasja, tako da smo, kjer se je pojavilo večglasje, uporabili samo najvišje tone.

Pri vseh eksperimentih smo v nadaljevanju uporabili enak postopek. Osemdeset odstotkov izbora pesmi smo uporabili za učenje kompozicionalnega modela. Množico pesmi za učenje smo izbrali tako, da smo za vsak variantni tip naključno izbrali osemdeset odstotkov pesmi. Model smo priredili tako, da namesto vzorcev, zapisanih v obliki pojavitvenih časov in tonskih višin, na izhodu dobimo zaporedje magnitud aktivacij vseh delov hierarhije. Na izhodnih podatkih smo s splošno tehniko samovzorčenja *(ang. bootstrap aggregating* ali *bagging)* na osnovi odločitvenih dreves zgradili odločitveni model za klasifikacijo v variantne tipe. Na zgrajeni hierarhiji smo izračunali rezultate na testni množici in jih z zgrajenim odločitvenim modelom klasificirali.

6.2 Rezultati in analiza napak

Ko smo klasificirali testne podatke, smo delovanje ocenili z izračunom točnosti:

$$A = \frac{\text{št. pravilno klasificiranih skladb}}{\text{št. vseh skladb}} \tag{6.1}$$

Rezultati, ki smo jih dobili za zgoraj opisane eksperimente, so prikazani v tabeli 6.1. Analiza teh rezultatov je opisana v nadaljevanju.

Tabela 6.1: Tabela prikazuje točnost klasifikacije pri treh različnih preizkusih.

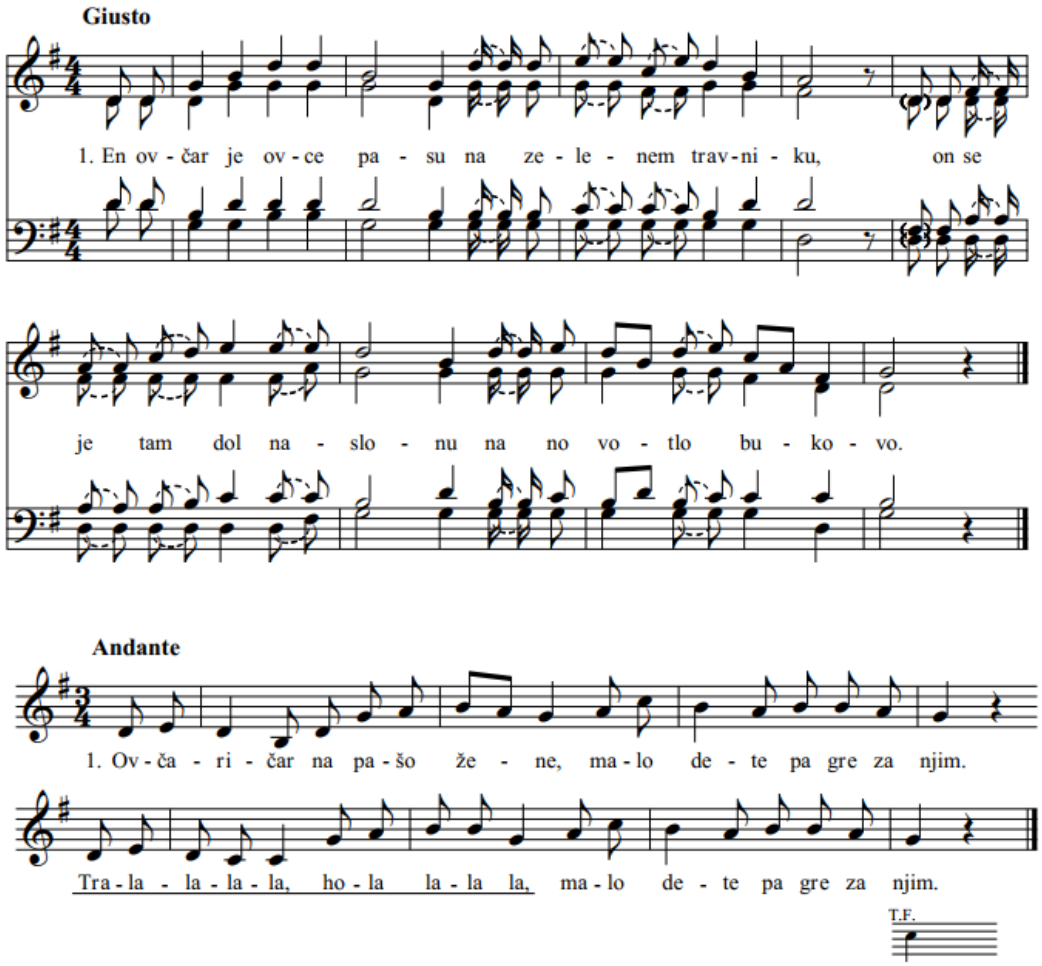
	Eksperiment 1	Eksperiment 2	Eksperiment 3
A	0.22	0.25	0.34

6.2.1 Analiza prvega eksperimenta

Pri osnovnem preizkusu smo dosegli nizko točnost napovedi. Po pregledu rezultatov smo odkrili štiri glavne probleme, zaradi katerih klasifikacija deluje slabo.

1. Problem besedila Ker naš model išče ponavljanja na podlagi melodije in ne besedila, odkrije in pravilno uvrsti samo tiste variante, ki so si podobne po melodiji, medtem ko so v zbirki OSNP pod istimi variantnimi tipi vključene tudi pesmi, ki so si podobne le v besedilih oz. v temi besedila, melodije pa so si tako različne, da ne najdemo pomembnih ponavljajočih vzorcev. Primer vidimo na sliki 6.1.

2. Večglasje in enoglasje Ker model na večglasnih skladbah ponavadi poišče večglasne vzorce, nastane problem, ko primerjamo večglasno in enoglasno skladbo. Pri enoglasni skladbi model vrne ponavljajoč enoglasni vzorec, pri večglasni skladbi pa ponavljajoč večglasni vzorec. Vzorca tako nista zaznana kot enaka - ne aktivirata istega dela v hierarhiji - in pri klasifikaciji,



Slika 6.1: Na sliki sta prikazani dve skladbi iz zbirke OSNP, ki sodita v isti variantni tip. Vidimo, da se po melodiji ne ujemata, saj sta skladbi razvrščeni v isti variantni tip zaradi tematike besedila.

kjer klasificiramo na podlagi aktiviranih delov, takih skladb ne klasificiramo v isti razred. Primer je prikazan na sliki 6.2.

3. Pomanjkljivosti baze Problem, ki smo ga odkrili, so tudi povsem različne skladbe, ki so v zbirki OSNP uvrščene v enak variantni tip. Ne vemo, ali gre za uvrščanje v isti variantni tip po kakšnem nam neznanem vidiku, definitivno pa so taki zapisi, ki niso redki, vprašljivi in bi bilo potrebno preveriti, ali gre za napake ali za kaj drugega. To so skladbe, ki se ne ujemajo ne po besedilu in ne po melodiji. Naš model jih posledično večinoma napačno klasificira. Tak primer vidimo na sliki 6.3.

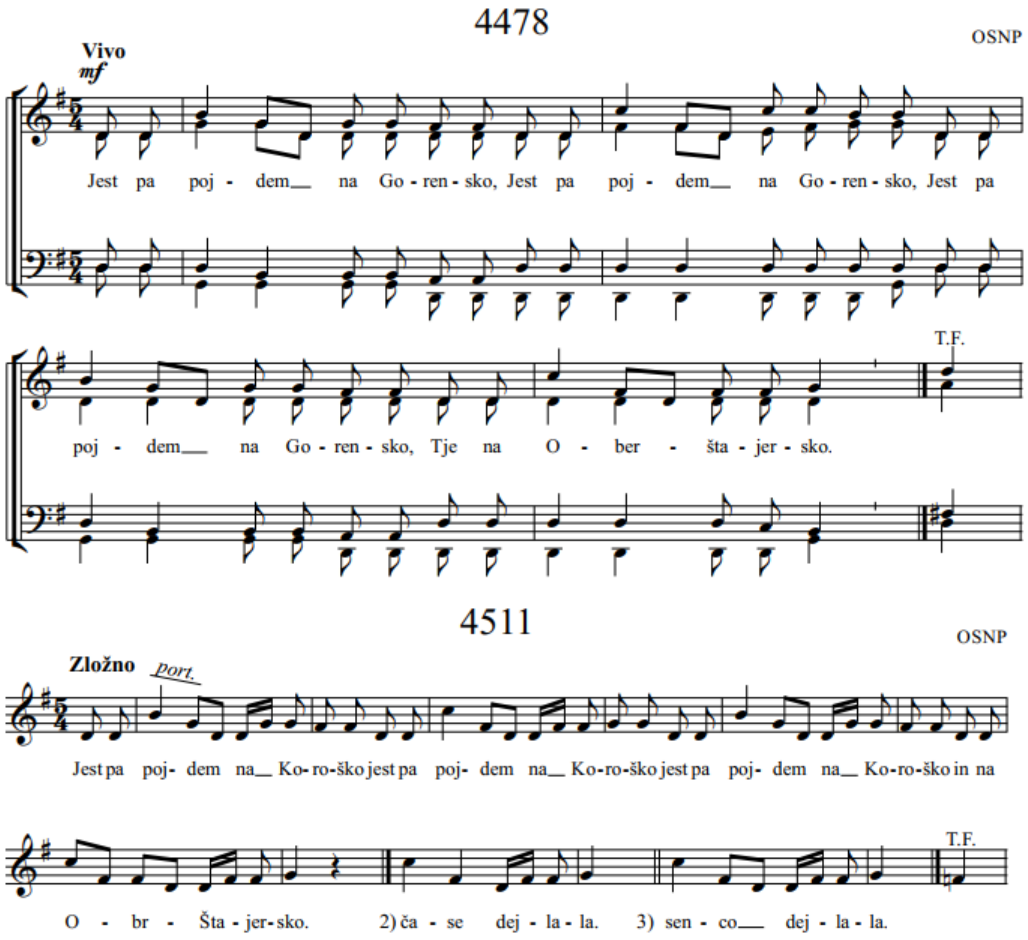
4. Enostavnost ljudske glasbe Naš model skladbe napačno razvršča tudi zaradi enostavnosti ljudskih melodij, ki so si posledično lahko tudi zelo podobne. Tako melodične podobnosti najdemo tudi v skladbah, ki pripadajo različnim variantnim tipom, kar prikazujemo na sliki 6.4. Če so si skladbe iz različnih variantnih tipov v melodiji podobne, jih naš model posledično napačno uvrsti v enak variantni tip.

6.2.2 Analiza drugega eksperimenta

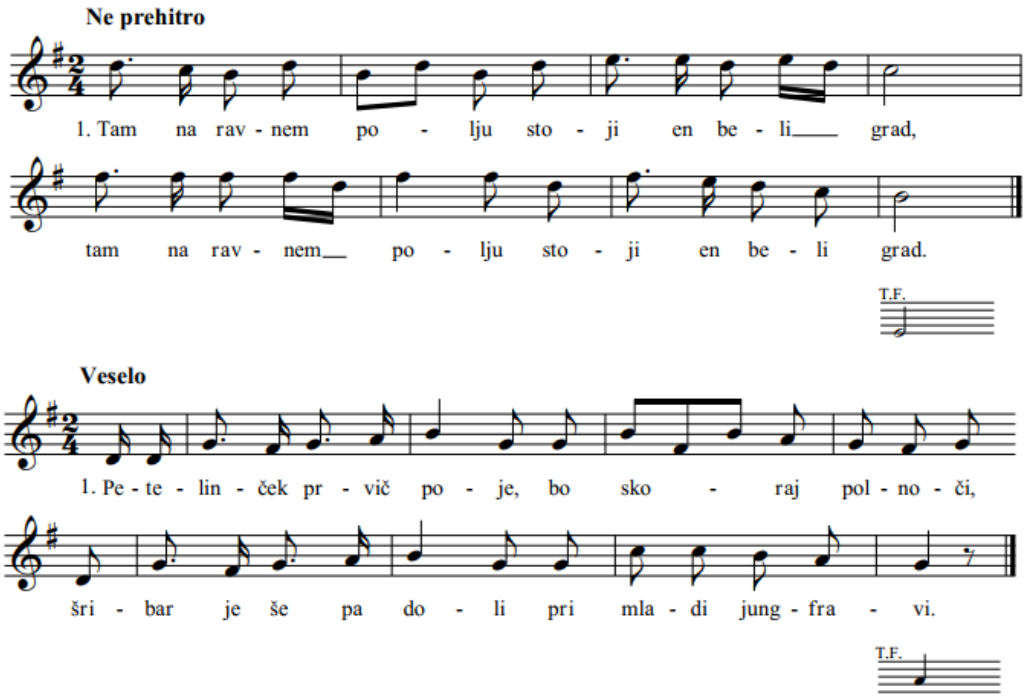
Zaradi problemov, ki nastanejo zaradi večglasja, smo klasifikacijo preizkusili samo na večglasnih skladbah, čeprav zaradi različnih vhodnih podatkov rezultatov ne bomo mogli neposredno primerjati. Pri eksperimentu smo prišli do zelo podobnih problemov kot prej. V zbirki so namreč večglasne skladbe včasih dvoglasne, včasih pa štiriglasne, kar je spet povzročilo sprožanje različnih delov hierarhije. Kljub vsemu se je natančnost malo dvignila.

6.2.3 Analiza tretjega eksperimenta

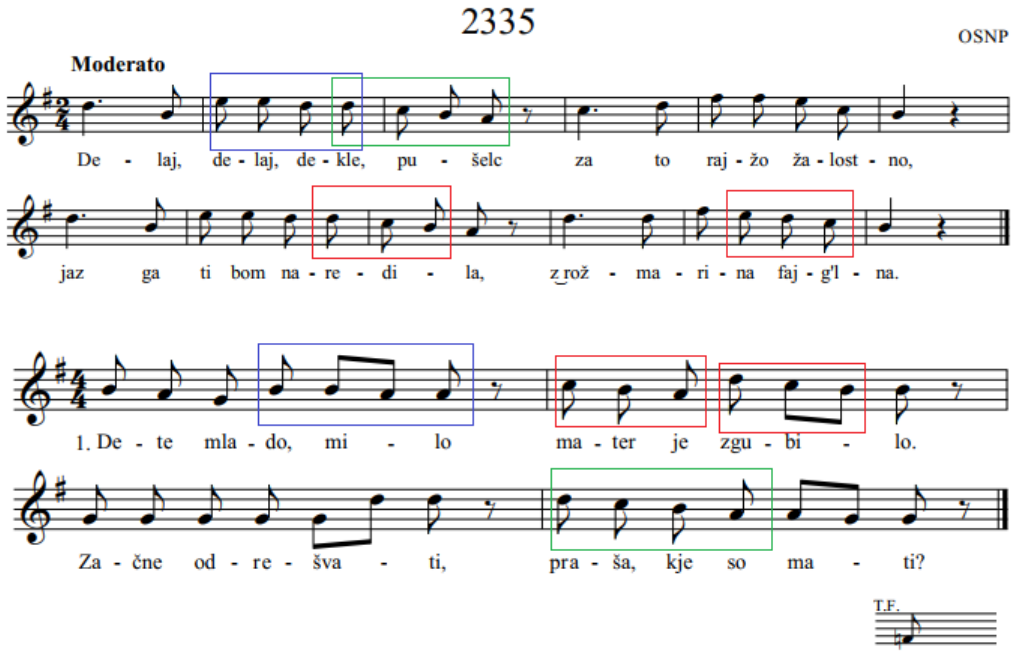
Te rezultate lahko primerjamo s prvim eksperimentom. Vidimo, da se točnost napovedi izboljša. Na ta način smo premostili problem večglasja in enoglasja pri iskanju enoglasnih melodij znotraj večglasja. Kljub vsemu so napačne



Slika 6.2: Na sliki je primer dveh skladb iz zbirke OSNP, ki sodita v isti variantni tip. Napačna razvrstitev je posledica tega, da se v zgodnjem primeru aktivirajo večglasni deli in se torej aktivacije razlikujejo od spodnjega primera, ki je enoglasen.



Slika 6.3: Na sliki sta prikazani dve skladbi iz zbirke OSNP, ki sodita v isti variantni tip. Ne ujemata se niti po besedilu in niti po melodiji. Predvidevamo, da so v zbirki OSNP napake.



Slika 6.4: Na sliki sta prikazani dve skladbi iz zbirke OSNP, ki sodita v različen variantni tip. Naš model ju na podlagi klasifikacije razvrsti v isti variantni tip. Z barvnimi okvirji so označeni vzorci, ki se pojavljajo v obeh skladbah. Zaradi takih podobnosti nekatere skladbe razvrstimo narobe.

klasifikacije zaradi problemov besedila, pomanjkljivosti zbirke in podobnosti slovenske ljudske glasbe ostale.

6.2.4 Diskusija

Pokazali smo, da so vzorci, ki jih dobimo z modelom, uporabni za nadaljnja opravila na področju pridobivanja informacij iz glasbe. Na podlagi rezultatov smo odkrili pomanjkljivosti v bazi OSNP. Kljub temu, da je točnost klasifikacije nizka, smo v zadnjem eksperimentu tretjino pesmi klasificirali pravilno. Za nadaljnje delo bi bilo na podlagi rezultatov modela v sodelovanju z muzikologi vsekakor zanimivo pregledati zbirko. Tako bi lahko prišli do ugotovitve, do kakšne mere model pravilno razvršča pesmi v razrede glede na podobnost v melodiji.

Poglavje 7

Zaključek

V delu smo pregledali pristope za pridobivanje ponavljajočih vzorcev iz polifonične simbolične glasbe. S kompozicionalnim hierarhičnim modelom smo razvili pristop za reševanje tega problema. Rezultate modela smo upodobili s spletno aplikacijo. Primerjali smo se z ostalimi algoritmi, ki so bili razviti in rezultate modela analizirali. Komentirali smo probleme in pomanjkljivosti, ki so prisotne pri tem opravilu. Model smo preizkusili na zbirki OSNP. Odkrili smo pomanjkljivosti zbirke in pokazali, da model lahko služi za iskanje podobnosti med skladbami.

Za nadaljnje delo bi bilo zagotovo potrebno vključiti raziskovalce, ki se ukvarjajo s kognitivno znanostjo in muzikologijo, da bi delo na interdisciplinarnem področju zares postalo interdisciplinarno. Predvsem bi bilo potrebno še bolj raziskati, kaj je zaznavno in analitično pomemben glasbeni vzorec in na podlagi tega v model vključiti dodatne mehanizme, ki bi med množico najdenih vzorcev bolje izbirali pomembne vzorce. Pri našem modelu bi v prvi vrsti potrebovali mehanizem, ki bi znotraj večglasnih vzorcev zaznal pomembne enoglasne linije, potem pa tudi mehanizme, ki bi pravilno prepoznavali začetke in konce pomembnih vzorcev in malo bolj muzikološko podprte mehanizme iskanja variacij. Potrebno bi bilo ustvariti večjo in boljšo zbirko ekspertnih anotacij za preizkušanje rešitev na tem področju.

Model vrača veliko zanimivih vzorcev, za izbor pomembnih ponavljajočih

vzorcev in za definicijo pomembnih ponavljajočih vzorcev pa nam ostaja še veliko prostora za nadaljevanje tega dela.

Literatura

[1] M. Schedl, E. Gómez, J. Urbano, Music information retrieval: Recent developments and applications, Foundations and Trends® in Information Retrieval 8 (2-3) (2014) 127–261. doi:10.1561/1500000042.
URL http://dx.doi.org/10.1561/1500000042

[2] J. S. Downie, Music information retrieval, Annual Review of Information Science and Technology 37 (1) (2003) 295–340. doi:10.1002/aris.1440370108.
URL http://dx.doi.org/10.1002/aris.1440370108

[3] N. Cook, A guide to musical analysis, Oxford University Press, USA, 1994.

[4] D. Meredith, Music analysis and point-set compression, Journal of New Music Research 44 (3) (2015) 245–270.

[5] G. Velarde, D. Meredith, T. Weyde, A wavelet-based approach to pattern discovery in melodies, in: Computational Music Analysis, Springer, 2016, pp. 303–333.

[6] M. Pesek, U. Medvešek, A. Leonardis, M. Marolt, Symchm: a compositional hierarchical model for pattern discovery in symbolic music representations, 11th Annual Music Information Retrieval eXchange (MIREX'15) (2015) 1–3.

[7] O. Lartillot, Submission to mirex discovery of repeated themes and sections, 10th Annual Music Information Retrieval eXchange (MIREX'14), Taipei.

[8] T. Collins, Discovery of repeated themes & sections - Music information retrieval exchange Wiki (2015).

[9] M. Pesek, A. Leonardis, M. Marolt, A compositional hierarchical model for music information retrieval, in: Proceedings of the 15th International Conference on Music Information Retrieval (ISMIR 2014), Citeseer, 2014, pp. 131–136.

[10] M. Pesek, A. Leonardis, M. Marolt, Pattern discovery and music similarity with compositional hierarchical model, in: CogMIR : August 12, 2016, Columbia University, New York, 2016, p. 8.

[11] T. Collins, Discovery of repeated themes and sections, Retrieved 20th July, http://www. musicir. org/mirex/wiki/2017: Discovery of Repeated Themes & Sections.

[12] T. Collins, J. Thurlow, R. Laney, A. Willis, P. Garthwaite, A comparative evaluation of algorithms for discovering translational patterns in baroque keyboard works.

[13] J.-L. Hsu, C.-C. Liu, A. L. Chen, Discovering nontrivial repeating patterns in music data, IEEE Transactions on Multimedia 3 (3) (2001) 311–325.

[14] I. Knopke, F. Jürgensen, A system for identifying common melodic phrases in the masses of palestrina, Journal of New Music Research 38 (2) (2009) 171–181.

[15] E. Cambouropoulos, M. Crochemore, C. S. Iliopoulos, M. Mohamed, M.-F. Sagot, A pattern extraction algorithm for abstract melodic representations that allow partial overlapping of intervallic categories, in:

Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005), 2005, pp. 167–174.

[16] S.-C. Chiu, M.-K. Shan, J.-L. Huang, H.-F. Li, Mining polyphonic repeating patterns from music data using bit-string based approaches, in: Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on, IEEE, 2009, pp. 1170–1173.

[17] C. Meek, W. P. Birmingham, Automatic thematic extractor, Journal of Intelligent Information Systems 21 (1) (2003) 9–33.

[18] H. Barlow, S. Morgenstern, A dictionary of musical themes, Crown Pub, 1975.

[19] D. Conklin, C. Anagnostopoulou, Representation and discovery of multiple viewpoint patterns, in: Proceedings of the International Computer Music Conference, 2001, pp. 479–485.

[20] D. Meredith, K. Lemström, G. A. Wiggins, Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music, Journal of New Music Research 31 (4) (2002) 321–345. doi:10.1076/jnmr.31.4.321.14162.
URL http://www.tandfonline.com/doi/abs/10.1076/jnmr.31.4.321.14162

[21] G. Peeters, Sequence representation of music structure using higher-order similarity matrix and maximum-likelihood approach., in: ISMIR, 2007, pp. 35–40.

[22] O. Nieto, M. M. Farbood, MIREX 2014 entry: Music segmentation techniques and greedy path finder algorithm to discover musical patterns (2014).

[23] S. Fidler, A. Leonardis, Towards scalable representations of object categories: Learning a hierarchy of parts, in: Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, IEEE, 2007, pp. 1–8.

[24] M. Pesek, Prepoznavanje akordov s hierarhičnim kompozicionalnim modelom: diplomsko delo, Ph.D. thesis, Univerza v Ljubljani, Fakulteta za računalništvo in informatiko (2012).
URL http://eprints.fri.uni-lj.si/1601/1/Pesek1.pdf

[25] M. Žerovnik, Kompozicionalni hierarhični model za ocenjevanje osnovnih frekvenc: diplomsko delo, Ph.D. thesis, Univerza v Ljubljani, Fakulteta za računalništvo in informatiko (2014).
URL http://eprints.fri.uni-lj.si/2657/

[26] L. Zakrajšek, Vizualizacija in razvoj analitičnega orodja s kompozicionalnim hierarhičnim modelom.
URL https://repozitorij.uni-lj.si/IzpisGradiva.php?lang=slv&id=80168

[27] U. Juvan, Iskanje ponavljajočih tem in vzorcev v glasbi s pomočjo kompozicionalnega hierarhičnega modela.
URL https://repozitorij.uni-lj.si/IzpisGradiva.php?lang=slv&id=84652

[28] D. Meredith, The ps13 pitch spelling algorithm, Journal of New Music Research 35 (2) (2006) 121–159.

[29] T. Collins, Improved methods for pattern discovery in music, with applications in automated stylistic composition (August 2011).
URL http://oro.open.ac.uk/30103/